

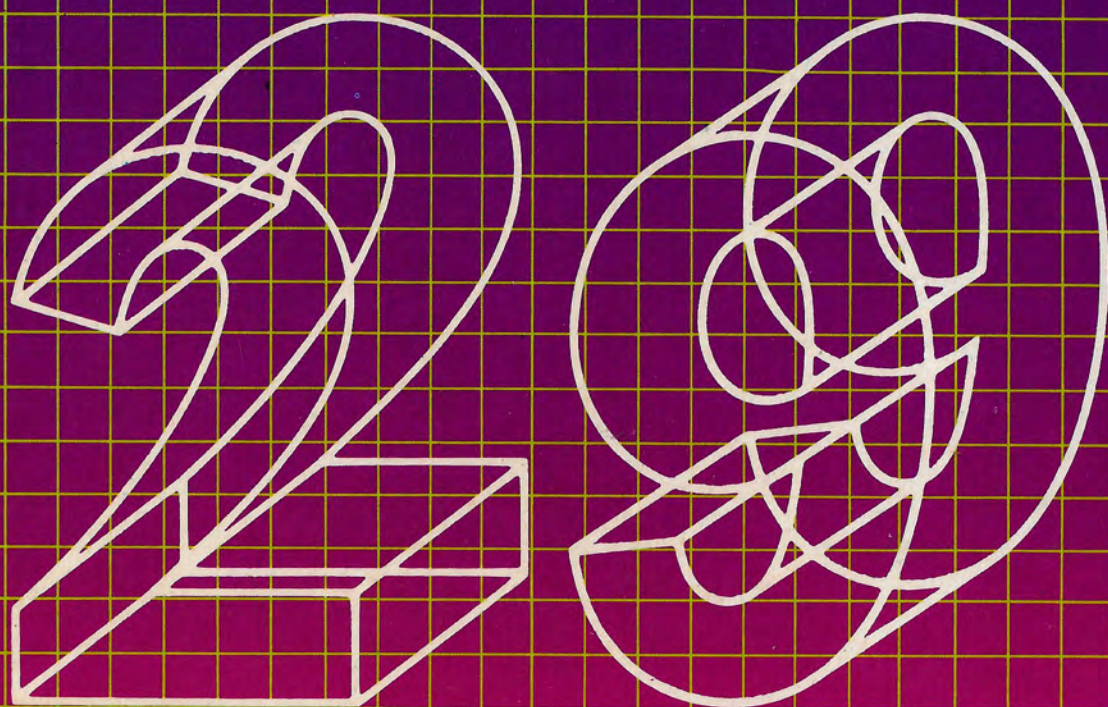
BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,

IBM, SPECTRUM, COMMODORE Y MSX



PON A TRABAJAR A TU ORDENADOR

HAZ COSAS UTILES CON EL

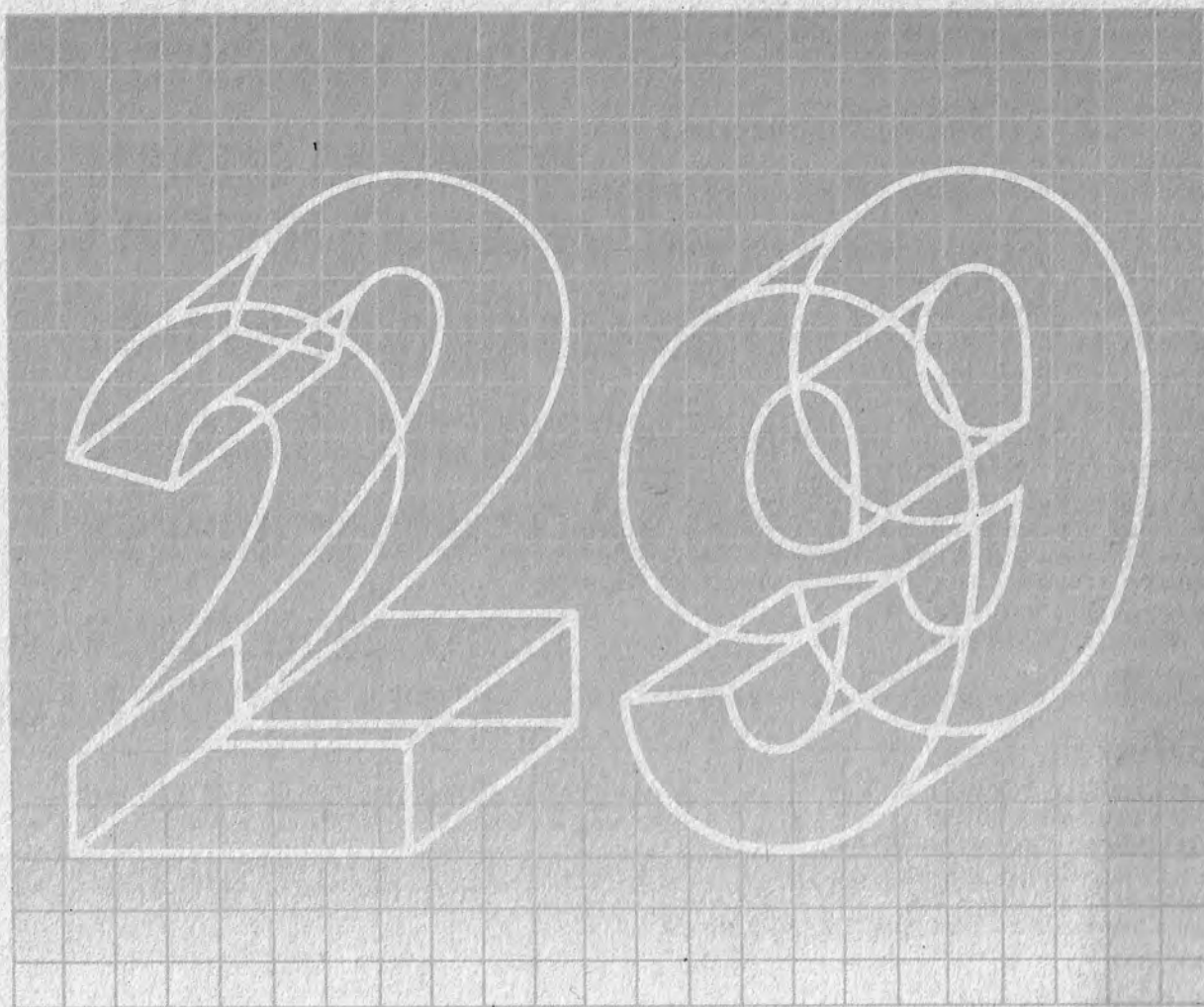
EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Director de la colección:

JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:

BRAVO-LOFISH.

Maquetación:

D. S. M.

Dibujos:

JOSE OCHOA

Tomo 29. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Jesús Bocho, Licenciado en Informática; Alejandro Marcos, Licenciado en Química. «El Taller de Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Publicidad:

Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.

Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-151-7

ISBN de la obra: 84-7688-047-2

Fotocomposición:

ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:

GRAFICAS EMA. (Madrid)

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-43.185-1986

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S. A.

Pedro Teixeira, 8-2.º. Teléf. 810 52 13. 28020 Madrid.

Agosto, 1987

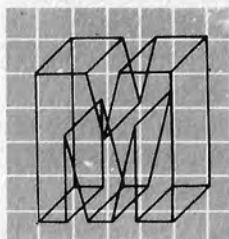
P.V.P. Canarias: 355.—Ptas.

INDICE

■ EXPERIENCIA Y PRACTICAS EN LOGO	5
■ MANEJO DE SPRITES Y ELEMENTOS GRAFICOS	15
■ TRUCOS Y RUTINAS BASICAS	25
■ EL TALLER DEL HARDWARE	33
■ APRENDER CON EL ORDENADOR	39
■ PEQUEÑA HISTORIA DE LA INFORMATICA	45
■ TEMAS MONOGRAFICOS DE VANGUARDIA	49
■ TERMINOLOGIA	55
■ VOCABULARIO DE INFORMATICA	57

EXPERIENCIA Y PRACTICAS EN LOGO

Operaciones con objetos



ENOSPRIMERO objeto en forma abreviada:

MP objeto

Devuelve el objeto especificado sin su primer elemento.

Si objeto es una palabra, suprime el primer carácter.

? ESCRIBE MP "CARBON
ARBON

? HAZ "A MP "SIEMPRE

? ESCRIBE :A
IEMPRE

? ESCRIBE MP "A

En la pantalla no aparece nada escrito, pues el primer carácter del objeto especificado, la letra A, es el primer elemento del objeto.

? ES MP "
MENOSPRIMERO NO ACEPTA COMO DATO

La primitiva MENOSPRIMERO no acepta la palabra vacía como dato.

Si objeto es una lista, la primitiva MENOSPRIMERO suprime la primera palabra o la primera lista.

? ES MP [TODAS AQUELLAS CASAS]
AQUELLAS CASAS

? ES MP [HOLA [ADIOS SIEMPRE]]
[ADIOS SIEMPRE]

? ES MP [A]

Al suprimir el primer elemento de la lista especificada no queda nada que escribir.

? ES MP []
MENOSPRIMERO NO ACEPTA []
COMO DATO

Esta primitiva podemos combinarla con otras, por ejemplo, con la primitiva PRIMERO.

? ES PRIMERO MENOSPRIMERO [HE
COMPRADO UN ORDENADOR]
COMPRADO

El primer elemento de la lista especificada sin su primer elemento, es la palabra COMPRADO.

El resultado es el mismo que si lo hubiésemos hecho por separado:

? HAZ "B [HE COMPRADO UN ORDENADOR]

? HAC "C MENOSPRIMERO :B

? ESCRIBE :C ESCRIBE PRIMERO :C

COMPRADO UN ORDENADOR
COMPRADO

En la pantalla aparece el contenido de la variable C y el primer elemento de C, que es el mismo que habíamos obtenido antes.

También podemos combinar la primitiva MP consigo misma.

? ES MP MP [LA FIESTA ES A LAS OCHO]
ES A LAS OCHO

Hemos obtenido una lista que carece de los dos primeros elementos de la lista especificada.

— PONPRIMERO objeto lista

A los objetos Logo podemos quitarles elementos.

EXPERIENCIA Y PRACTICAS EN LOGO

en forma abreviada:

PP objeto lista

Devuelve una lista en la que se incluye el objeto especificado, como primer elemento, y la lista especificada.

? ES PP "LOS [GUANTES BLANCOS]
LOS GUANTES BLANCOS

? ES PP [TODOS] [SE HAN IDO]
[TODOS] SE HAN IDO

? ES PP [SUELO] []
[SUELO]

? ES PP [] [ES CORRECTO]
[] ES CORRECTO

? ES PP "A "INCORRECTO
PONPRIMERO NO ACEPTA IN-
CORRECTO COMO DATO

El primer dato que acompaña a la primitiva PONPRIMERO puede ser cualquier objeto LOGO (un carácter, una palabra o una lista), el segundo dato debe ser una lista.

Podemos combinar esta primitiva con otras, como hacíamos con MENOSPRIMERO, por ejemplo, con la primitiva PRIMERO.

? PARA Q1
> HAZ "B PRIMERO PP "YO [HE COM-
PRADO UN ORDENADOR]
> ES :B
> FIN

Si ejecutamos este procedimiento:

? Q1

obtenemos la palabra YO, que es el primer elemento de la lista obtenida por la primitiva PP con objeto y la lista especificada.

En el siguiente procedimiento realizamos lo mismo, pero separando las primitivas PP y PRIMERO.

? PARA Q
> HAZ "B PP "YO [HE COMPRADO UN
ORDENADOR]
> HAZ "C PRIMERO :B ES :C
> FIN

? Q
YO

El siguiente ejemplo utiliza dos veces seguidas la primitiva PP.

? PARA P

> ES PP PP "LAS [CASAS NO TIENEN
VENTANAS] [NI BALCONES]
> FIN

Si lo ejecutamos:

? P

obtenemos:

[LAS CASAS NO TIENEN VENTANAS]
NI BALCONES

Si para clarificarnos queremos separarlas:

? PARA P1
> HAZ "A PP "LAS [CASAS NO TIE-
NEN VENTANAS]
> HAZ "B PP :A [NI BALCONES]
> ES :B
> FIN

Si lo ejecutamos, obtenemos lo mismo que con el procedimiento P.

? P1
[LAS CASAS NO TIENEN VENTANAS]
NI BALCONES

NOTA: La sintaxis de esta primitiva en el LOGO del SPECTRUM es:

PONPRI objeto [lista]

— FRASE objeto1 objeto2

en forma abreviada:

FR objeto1 objeto2

La primitiva FRASE devuelve una frase formada por objeto1 y objeto2.

? ES FRASE "HOLA "JOSE
HOLA JOSE

? ES FRASE [HOLA] [JOSE]
HOLA JOSE

Recuerda que con la primitiva LISTA conseguimos algo parecido, si sus datos son listas escribe en la pantalla los corchetes de las listas.

? ES LISTA [HOLA] [JOSE]
[HOLA] [JOSE]

Si cuando utilices la primitiva FRASE quieres indicar más de dos objetos, debes encerrar a la primitiva y a los objetos entre paréntesis.

? ES FRASE "HOLA "A "TODOS

Construye frases uniando diversos objetos.

HOLA A
NO SE QUE DEBO HACER CON TODOS

El LOGO ha mostrado un mensaje de error, hemos indicado tres objetos sin encerrarlos, junto con la primitiva, entre paréntesis.

? ES (FRASE "HOLA "A "TODOS)
HOLA A TODOS

? ES (FRASE [HOLA] "A [TODOS])
HOLA A TODOS

? ES (FRASE [HOLA] "Y [ADIOS A TODOS])
HOLA Y ADIOS A TODOS

■ Espacio de memoria

La memoria del ordenador es un soporte físico y en él se van almacenando los procedimientos que vamos definiendo.

Esta memoria está organizada en pequeñas celdas (nodos), pudiendo almacenar cada una de ellas cinco octetos (1 octeto es igual a un byte, un byte es igual a ocho bits). Cada una de estas celdas es la unidad de partición del área de trabajo del Logo.

Cada procedimiento que definimos, así como la ejecución de éstos por el intérprete Logo, van ocupando celdas de memoria y de esta forma reduciendo el espacio libre que tenemos.

Cada letra, palabra, procedimiento, etcétera, ocupan un número determinado de estas celdas.

Si definimos un gran número de procedimientos, la memoria del ordenador puede llegar a llenarse e impedir el seguir trabajando, ya que no existiría espacio libre para ello.

A medida que vamos trabajando, en muchas celdas se va almacenando información que no es necesaria durante todo el tiempo que dura la sesión de trabajo. En definitiva, esta información lo que hace es ocupar un lugar innecesario y por ello el Logo de vez en cuando realiza, de una manera automática, una limpieza de estas celdas, liberándolas de la información que contienen y ganando de esta forma más espacio libre. Intenta en todas

las ocasiones liberar el mayor espacio posible de la memoria.

En realizar este proceso se tarda un determinado tiempo.

En varias ocasiones habrás observado que cuando se está ejecutando un determinado procedimiento, hay una pequeña pausa en la cual la ejecución del procedimiento se detiene. Esta pausa es la que se utiliza para realizar la limpieza.

El Logo puede también realizar este mismo proceso cuando nosotros deseemos, para lo que cuenta con la primitiva:

? RECICLA

Al introducir esta orden se liberarán el mayor número de nodos posibles.

Es conveniente introducir esta orden antes de ejecutar un proceso y evitar de esta forma que existan interrupciones a lo largo de su ejecución.

El Logo dispone de otra orden que nos da el número de nodos libres que tiene el área de trabajo de la memoria del ordenador. Se trata de la primitiva:

? NODOS

Una vez que hemos introducido esta orden, sabremos el espacio que nos queda en memoria para poder seguir trabajando, definiendo procedimientos, ejecutándolos, etc.

También podemos saber el espacio (en nodos) que ocupa un determinado procedimiento. Para ello lo que tenemos que hacer es:

— Una vez cargado el Logo introducimos la orden NODOS. Obtendremos un número que nos indicará el espacio libre que tenemos cuando la memoria todavía está vacía. Anota el número que nos ha devuelto.

— A continuación define un procedimiento.

— Si vuelves a introducir la orden NODOS, obtendrás otro número que será menor al anterior. Este número indicará la memoria que hay libre después de haber realizado un determinado trabajo.

La diferencia entre estos dos números nos dará la cantidad de nodos que hemos utilizado para definir el procedimiento.

Cada palabra que escribimos en Logo se almacena una sola vez en la memoria, es

Si le quitamos el primer elemento a una palabra, le estamos quitando un carácter.

EXPERIENCIA Y PRACTICAS EN LOGO

decir, si utilizamos, por ejemplo, el mismo nombre para un procedimiento y para una variable, éste sólo se almacenará una vez y el Logo controlará y diferenciará cuál corresponde en cada caso.

Una primitiva que nos permite conocer todas las palabras que se han utilizado es:

? CONTENIDO

En la versión del Logo para los MSX, esta orden devolverá una lista con todas las palabras que se han utilizado como nombres de procedimientos, de variables, etc.

Al comenzar una sesión, la orden CONTENIDO no devuelve nada.

Si introducimos la orden BOTODO (BORra TODO), todas las palabras desaparecen de CONTENIDO.

En la versión del Logo para los PC-COMPATIBLES, esta orden devolverá una lista con todas las palabras que se han utilizado como nombres de procedimientos, de variables, etc., así como la lista de todas las primitivas de que dispone.

Al comenzar una sesión, la orden CONTENIDO devuelve una lista con todas las primitivas Logo.

En la versión del Logo para el Spectrum, las primitivas:

? RECICLA

y

? NODOS

son iguales.

La primitiva CONTENIDO no tiene la misma función que hemos visto en los casos anteriores.

En este caso la sintaxis es:

? CONTENIDO "Nombre de variable

Esta orden devuelve el contenido de la variable que se especifica en Nombre de Variable.

■ Cuadro resumen

— MENOSPRIMERO objeto

o

MP objeto

Devuelve el objeto especificado sin su primer elemento.

— POMPRIMERO objeto lista

o

PP objeto lista

Devuelve una lista formada por la lista especificada precedida del objeto.

NOTA: La sintaxis en el SPECTRUM es:

PONPRI objeto [lista]

— FRASE objeto1 objeto2

o

FR objeto1 objeto2

La primitiva FRASE devuelve una frase formada por objeto1 y objeto2.

El número de datos puede ser mayor de dos, en esos casos la primitiva FRASE y sus datos deben ir entre paréntesis.

(FRASE objeto1 objeto2 objeto3 ...)

— RECICLA

Se liberan de la memoria del ordenador el mayor número de nodos posibles que contengan información innecesaria.

— NODOS

Devuelve un número que indica la cantidad de nodos libres de que dispone la memoria en el momento que se introduce la orden.

— CONTENIDO

MSX:

Devuelve una lista con todas las palabras que se han utilizado para nombres de procedimientos, de variables, etc.

PC-COMPATIBLES:

Devuelve una lista con todas las palabras que se han utilizado para nombres de procedimientos, de variables, etc., y todas las primitivas Logo.

SPECTRUM:

CONTENIDO "nombre de variable

Devuelve el valor asignado a la variable que se especifica.

A cualquier lista le podemos añadir en la primera posición el objeto que queramos.

Ejercicios

1. Diseña a través de los datos que introduzcas en el ordenador una tarjeta del tipo:

NOMBRE Y APELLIDOS
 PROFESION
 DIRECCION
 TELEFONO
 CODIGO POSTAL Y PROVINCIA

Para hacer una tarjeta de este tipo deberás ejecutar primero los procedimientos que hayas definido para introducir los datos.

2. Realiza el siguiente dibujo:

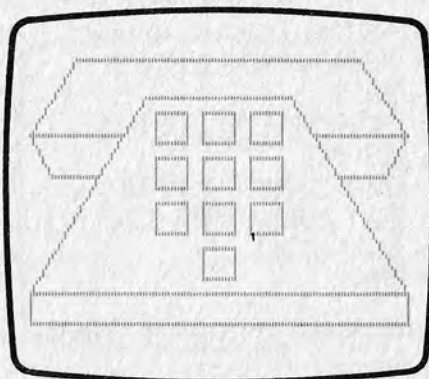


Fig. 1.

3. Realiza el siguiente dibujo:

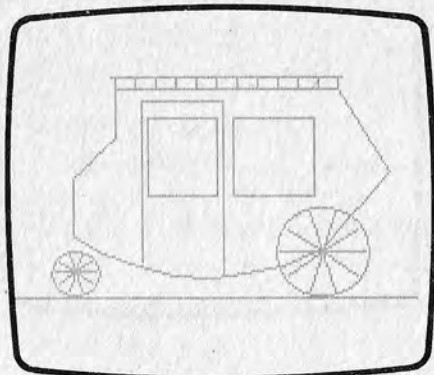


Fig. 2.

4. Define los procedimientos necesarios para realizar el siguiente proceso:

1. Pedir que se teclee una palabra de cuatro caracteres.

2. Los caracteres tecleados se escribirán por el orden que se han introducido en los puntos medios de los lados de la pantalla.

3. Desplazar hacia el centro de la pantalla, también en el orden en que se han introducido, los cuatro caracteres para formar la palabra.

4. Enmarcar la palabra que se ha formado.

5. ¿Son correctas las siguientes órdenes?:

- ? ES PP "SOLO "AGUA
- ? HAZ "B LL
- ? HAZ "A PRIMERO :B
- ? HAZ "C ULTIMO :B
- ? SI :A = :C [ESCRIBE MP :B] [ESCRIBE PP :C :B]
- ? HAZ "A LL
- ? HAZ "B PP "HOLA :A
- ? HAZ "C MP :B
- ? ESCRIBE :C ESCRIBE :B
- ? HAZ "A O
- ? REPITE 10 [PONCURSOR LISTA 1 + :A 2 ES "A HAZ "A :A + 1]
- ? ES PP PP "TODO "ES [IGUAL]
- ? ES (FRASE "HACE "MUCHO [] "CALOR)

Solución a los ejercicios

1:

Primero definimos un procedimiento que presenta en pantalla los conceptos que deben introducirse para poder realizar la tarjeta, una vez presentados se hace una llamada al procedimiento DATOS.

- ```
? PARA CABE
> PT
> BT
> PONCURSOR [10 1]
> ESCRIBE "DATOS$ PERSONALES
> PONCURSOR [2 8]
> ESCRIBE "NOMBRE
> PONCURSOR [2 9]
> ESCRIBE "APELLIDO$ 1....
> PONCURSOR [2 10]
> ESCRIBE "APELLIDO$ 2....
```

**El segundo dato de la primitiva PONPRIMERO debe ser obligatoriamente una lista.**

## EXPERIENCIA Y PRACTICAS EN LOGO

```
> PONCURSOR [2 11]
> ESCRIBE "DIRECCION.....
> PONCURSOR [2 12]
> ESCRIBE "POBLACION.....
> PONCURSOR [2 13]
> ESCRIBE [CODIGO POSTAL.]
> PONCURSOR [2 14]
> ESCRIBE "TELEFONO.....
> PONCURSOR [2 15]
> ESCRIBE "PROFESION.....
> DATOS
> PONCURSOR [10 19]
> ESCRIBE "¿OTRO$ DATO?$ $(S$/N$)
> PONCURSOR [27 19]
> HAZ "X LEECAR
> SI :X = "N [BT PONCURSOR [1 23]
ALTO] [CABE]
> FIN
```

El procedimiento DATOS permite introducir los datos de todos los conceptos anteriormente presentados en la pantalla.

```
? PARA DATOS
> PONCURSOR [17 8]
> HAZ "N LL
> PONCURSOR [17 9]
> HAZ "A1 LL
> PONCURSOR [17 10]
> HAZ "A2 LL
> PONCURSOR [17 11]
> HAZ "D LL
> PONCURSOR [17 12]
> HAZ "P LL
> PONCURSOR [17 13]
> HAZ "C LL
> PONCURSOR [17 14]
> HAZ "T LL
> PONCURSOR [17 15]
> HAZ "PR LL
> HAZ "NO (FRASE :N :A1 :A2)
> HAZ "P FRASE :C :P
> TARJETA :NO :PR :D :T :P
> FIN
```

Después de la entrada de datos se realiza una llamada al procedimiento que presenta en pantalla la tarjeta.

Al procedimiento TARJETA le pasamos seis datos:

```
N nombre y apellidos
P profesión
D dirección
T teléfono
PO código postal y provincia
? PARA TARJETA :N :P :D :T :PO
> BT
> PONCURSOR [12 6]
> ES :N
> PONCURSOR [12 7]
> ES :P
> PONCURSOR [12 10]
> ES [C /]
> PONCURSOR [16 10]
> ES :D
> PONCURSOR [12 11]
> ES "Tel.
> PONCURSOR [17 11]
> ES :T
> PONCURSOR [12 12]
> ES :PO
> PONCURSOR [10 19]
> ES [PULSA UNA TECLA]
> HAZ "E LC
> FIN
```

Realicemos la primera tarjeta:

? CABE

en la pantalla aparece:

Fig. 3.

Introducimos ahora los datos que deseamos, por ejemplo:

***Cada cierto tiempo el Logo realiza una limpieza de la memoria liberando nodos cuya información no sea necesaria.***

DATOS PERSONALES

NOMBRE..... ESTHER  
 APELLIDO 1.... VIGO  
 APELLIDO 2.... MARTIN  
 DIRECCION..... FUENCARRAL, 4  
 POBLACION..... MADRID  
 CODIGO POSTAL. 28044  
 TELEFONO..... 435 67 81  
 PROFESION..... FARMACEUTICA

Fig. 4.

y obtendremos la tarjeta:

ESTHER VIGO MARTIN  
 FARMACEUTICA

C / FUENCARRAL, 4  
 Tel. 435 67 81  
 28044 MADRID

Fig. 5.

2:

Para realizar el dibujo del teléfono definimos el siguiente procedimiento:

```
? PARA TELEFONO
> BP
> SL
> PONPOS [-60 -60]
> PONRUMBO 0
> BL
> REPITE 2 [AV 10 GD 90 AV 120 GD
90]
> SL
> PONRUMBO 30
> PONY -50
> BL
> AV 75
> SL
> PONPOS [60 -50]
> PONRUMBO 330
```

```
> BL
> AV 75
> PONRUMBO 270
> AV 45
> PONRUMBO 210
> AV 30
> PONRUMBO 270
> AV 15
> PONRUMBO 330
> AV 15
> PONRUMBO 90
> AV 30
> RE 30
> PONRUMBO 30
> AV 30
> PONRUMBO 90
> AV 90
> PONRUMBO 150
> AV 30
> PONRUMBO 270 AV 30
> RE 30 GI 60
> AV 15
> PONRUMBO 270
> AV 15
> SL PONPOS [-20 10]
> PONRUMBO 90
> TECLAS
> SL PONPOS [-20 -5]
> TECLAS
> SL PONPOS [-20 -20]
> TECLAS
> SL PONPOS [-5 -35]
> BL REPITE 4 [AV 10 GD 90]
> FIN
```

En el procedimiento TELEFONO realizamos varias llamadas al procedimiento TECLAS.

```
? PARA TECLAS
> REPITE 3 [BL REPITE 4 [AV 10 GD
90] SL AV 15]
> FIN
```

Se encarga de dibujar los cuadrados que simulan los números de un teléfono. Lo ejecutamos tecleando:

```
? TELEFONO
```

3:

Para realizar el dibujo del carro vamos

**Podemos saber en un momento determinado la cantidad de memoria libre que tenemos disponible**

## EXPERIENCIA Y PRACTICAS EN LOGO

a definir procedimientos que dibujan diferentes partes de éste.

En primer lugar, definimos el que dibuja la rueda grande del carro:

```
? PARA RUEDA1
> SL
> PONPOS [50 -20]
> BL
> REPITE 18 [AV 8 GD 20]
> SL
> AV 4 GD 90
> AV 23
> GI 90
> BL
> HAZ "A 0
> REPITE 10 [BL AV 21 SL RE 21 GD
36]
> FIN
```

RUEDA 2, dibuja la rueda pequeña:

```
? PARA RUEDA2
> SL
> PONPOS [-70 -42]
> BL
> REPITE 18 [AV 4 GD 20]
> SL
> AV 2 GD 90
> AV 11
> GI 90
> HAZ "A 0
> REPITE 10 [BL AV 10 SL RE 10 GD
36]
> FIN
```

Para dibujar la cabina definimos el siguiente procedimiento:

```
? PARA CABINA
> SL
> PONPOS [55 -40]
> PONRUMBO 40
> BL
> AV 50
> PONRUMBO -30
> AV 50
> PONRUMBO 270
> AV 110
> PONRUMBO 180
> AV 25
> PONRUMBO 225
> AV 30
> PONRUMBO 180
```

```
> AV 30
> GI 50
> REPITE 7 [GI 9 AV 18]
> FIN
```

Las ventanas las dibuja:

```
? PARA VENTA
> SL
> PONPOS [5 -55]
> PONRUMBO 0
> BL
> AV 90
> GI 90
> AV 40
> GI 90
> AV 83
> SL
> PONPOS [2 -14]
> PONRUMBO 0
> BL
> REPITE 2 [AV 40 GI 90 AV 34 GI 90]
> SL
> PONPOS [10 -14]
> PONRUMBO 0
> BL
> REPITE 4 [AV 40 GD 90]
> FIN
```

Y, por último, definimos el procedimiento que dibuja la parte donde van las maletas:

```
? PARA MALE
> SL
> PONPOS [-48 43]
> PONRUMBO 0
> REPITE 12 [BL AV 4 RE 4 GD 90 SL
AV 10 GI 90]
> AV 4 GI 90
> AV 8
> BL
> REPITE 2 [AV 114 GD 90 AV 1 GD
90]
> FIN
```

Una vez que tenemos los procedimientos que dibujan las diferentes partes del carro, definimos el que inicializa el proceso y engloba a todos:

```
? PARA CARRO
> BP
> OT
> SL
```

***Una palabra sólo se almacena una vez en la memoria.***

```

> PONPOS [-100 -66]
> PONRUMBO 90
> BL
> AV 200
> RUEDA 1
> RUEDA 2
> CABINA
> VENTA
> MALE
> FIN

```

Para ejecutarlo bastará con introducir:

? CARRO

4:

En primer lugar, definimos el procedimiento que escribe en la pantalla el mensaje de petición de la palabra:

```

? PARA A
> PONCURSOR [1 1]
> ES [ESCRIBE UNA PALABRA
DE CUATRO LETRAS]
> FIN

```

El siguiente procedimiento es el que realiza la petición de cuatro caracteres y los escribe en los extremos de la pantalla:

```

? PARA B
> HAZ "A LC
> HAZ "B LC
> HAZ "C LC
> HAZ "D LC
> BT
> BP
> PONCURSOR [0 11]
> ES :A
> PONCURSOR [19 0]
> ES :B
> PONCURSOR [37 11]
> ES :C
> PONCURSOR [21 22]
> ES :D
> ESPERA 50
> FIN

```

Una vez que tenemos los caracteres de la palabra situados en el lugar adecuado, los desplazamos hasta formar la palabra en la posición central de la pantalla:

```

? PARA C
> HAZ "I 1

```

```

> REPITE 18 [PONCURSOR (LISTA
:I 11) ES :A HAZ "I :I + 1]
> HAZ "J 1
> REPITE 11 [PONCURSOR (LISTA
19 :J) ES :B HAZ "J :J + 1]
> HAZ "K 1
> REPITE 17 [PONCURSOR (LISTA
37 - :K 11) ES :C HAZ "K :K + 1]
> HAZ "H 1
> REPITE 11 [PONCURSOR (LISTA
21 22 - :H) ES :D HAZ "H :H + 1]
> FIN

```

Una vez que la palabra se ha formado, la enmarcamos con el siguiente procedimiento:

```

? PARA D
> SL
> PONPOS [-16 6]
> BL
> REPITE 2 [AV 10 GD 90 AV 31 GD
90]
> FIN

```

TODO inicializa el proceso y engloba a todos los procedimientos:

```

? PARA TODO
> BP
> PM
> OT
> BT
> A
> B
> C
> D
> FIN

```

Si lo ejecutamos:

? TODO

En la pantalla aparecerá:

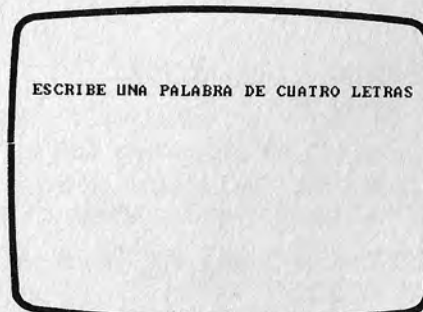


Fig. 6.

***Podemos conocer todas las palabras que se han utilizado en un proceso.***

## EXPERIENCIA Y PRACTICAS EN LOGO

Esta pantalla permanecerá hasta que tecleemos cuatro caracteres. Si, por ejemplo, pulsamos las teclas:

H  
O  
L  
A

En la pantalla obtendremos:

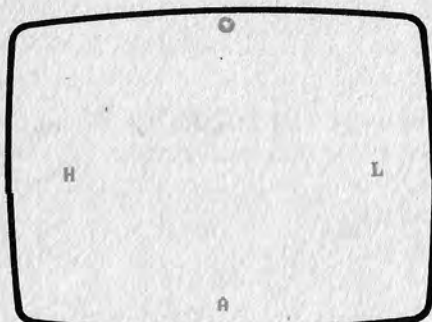


Fig. 7.

Al cabo de una pequeña pausa, las letras se empezarán a desplazar hasta situarse en el centro de la pantalla formando la palabra:

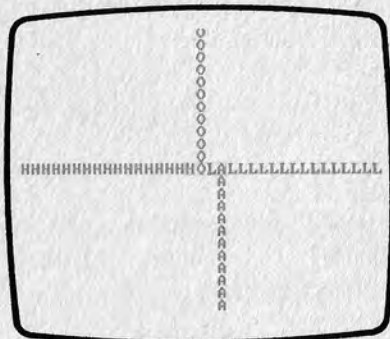


Fig. 8.

5:

— ? ES PP "SOLO "AGUA  
INCORRECTO.

La primitiva PP no acepta una palabra como segundo dato, sólo acepta listas.  
El LOGO devuelve el mensaje de error:

PONPRIMERO NO ACEPTA AGUA  
COMO DATO

— ? HAZ "B LL  
? HAZ "A PRIMERO :B

**La memoria del ordenador está organizada en nodos.**

? HAZ "C ULTIMO :B  
? SI :A = :C [ ESCRIBE MP :B ] [ ESCRIBE PP :C :B ]

CORRECTO.

Leemos una lista del teclado, almacenamos en la variable A el primer elemento y en C el último. Si el primer elemento es igual al último (:A = :C), en la pantalla aparece escrita la lista tecleada menos su primer elemento; si no son iguales, en la pantalla aparece la lista tecleada, incluyendo su último elemento como primer elemento.

— ? HAZ "A LL  
? HAZ "B PP "HOLA :A  
? HAZ "C MP :B  
? ESCRIBE :C ESCRIBE :B

CORRECTO.

La variable C contendrá la lista leída, la variable B la lista formada por la palabra HOLA como primer elemento y por la lista leída del teclado.

— ? HAZ "A 0  
? REPITE 10 [ PONCURSOR LISTA 1  
+ :A 2  
ES "A HAZ "A :A + 1 ]

CORRECTO.

La columna donde se posiciona el cursor va aumentando desde 1 hasta 10 y la línea será siempre la 2.

— ? ES PP PP "TODO "ES [ IGUAL ]  
INCORRECTO.

Recibiremos el mensaje:

PONPRIMERO NO ACEPTA ES COMO  
DATO

La forma correcta sería:

? ES P PP "TODO [ ES ] [ IGUAL ]

en la pantalla aparecerá:

[ TODO ES ] IGUAL

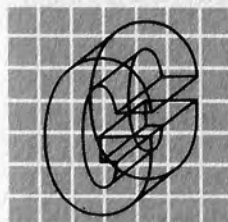
— ? ES (FRASE "HACE "MUCHO [ ]  
"CALOR)

CORRECTO.

En la pantalla aparecerá:

HACE MUCHO CALOR

# MANEJO DE SPRITES Y ELEMENTOS GRAFICOS



CONTINUANDO con el juego que hemos ido desarrollando últimamente para todos los ordenadores, a continuación damos la versión del mismo para el COMMODORE 64.

Como todos sabemos, el objetivo del juego es comerse el mayor número de puntos antes de que los fantasmas que habitan el laberinto donde se desarrolla el juego nos coman.

Para mover nuestro queso por la pantalla tendremos que usar las siguientes teclas:

O: Mover el quesito a la izquierda.  
P: Mover el quesito a la derecha.  
Q: Mover el quesito hacia arriba.  
A: Mover el quesito hacia la derecha.

Aparte de los puntos normales que aparecen en la pantalla, también podemos ver cuatro asteriscos. En la versión original de este juego, cuando el PAC-MAN se comía uno de los dichos puntos, los fantasmas cambiaban de color y entonces era el momento de comérnoslos. En esta nueva versión no nos podemos comer los fantasmas, pero al comernos los asteriscos, aparte de la bonificación de 50 puntos, nuestro PAC-MAN se moverá más de prisa y, por tanto, será más difícil que nos cojan los fantasmas.

```

1000 REM *****
1010 REM *
1020 REM * P A C - M A N *
1030 REM *
1040 REM * P A R A C O M M O D O R E *
1050 REM *
1060 REM *****
1070 REM
1080 REM *****
1090 REM *
1100 REM * A U T O R *
1110 REM * ----- *
1120 REM *
1130 REM * Fco. Morales Guerrero *
1140 REM *
1150 REM *****
1160 REM
1170 REM *****
1180 REM *
1190 REM * (c) Ed. Siglo Cultural *
1200 REM * (c) 1987. *
1210 REM *
1220 REM *****
1230 REM

```

## MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

1240 REM *****
1250 REM * INICIALIZACION DEL PROGRAMA *
1260 REM *****
1270 REM
1280 DIM N(9),N$(10),P(10)
1290 DIM S(25,41),M$(2,3),I(2,8)
1300 LET M$(1,0)=" "
1310 LET M$(1,1)="*"
1320 LET M$(1,3)=" "
1330 LET M$(2,0)=" "
1340 LET M$(2,1)="*"
1350 LET M$(2,3)=" "
1360 LET PA=1
1370 POKE 53281!,0
1380 POKE 53280!,0
1390 PRINT CHR$(30)
1400 PRINT CHR$(147)
1410 PRINT TAB(15);"COME - COCOS"
1420 PRINT TAB(14);"===== "
1430 PRINT CHR$(28)
1440 POKE 211,9:POKE 214,13:PRINT "CARGANDO DATAS"
1450 PRINT CHR$(31)
1460 POKE 211,11:POKE 214,11:PRINT "ESPERA UN MOMENTO."
1470 REM
1480 REM *****
1490 REM * LECTURA DE LOS CARACTERES *
1500 REM *****
1510 REM
1520 FOR I=0 TO 9
1530 READ N(I)
1540 NEXT I
1550 REM
1560 REM *****
1570 REM * LECTURA DE LOS RECORDS *
1580 REM *****
1590 REM
1600 FOR I=1 TO 10
1610 READ N$(I),P(I)
1620 NEXT I
1630 REM
1640 REM *****
1650 REM * LECTURA DE LAS DIRECCIONES *
1660 REM *****
1670 REM
1680 FOR I=1 TO 2
1690 FOR J=1 TO 8
1700 READ I(I,J)
1710 NEXT J
1720 NEXT I
1730 REM
1740 REM *****
1750 REM * PROGRAMA PRINCIPAL *
1760 REM *****
1770 REM
1780 PRINT CHR$(147)
1790 GOSUB 5410:REM PRESENTACION
1800 GOSUB 4600:REM TABLA DE RECORDS
1810 GOSUB 5170:REM INSTRUCCIONES
1820 GOSUB 5310:REM PULSA UNA TECLA
1830 GOSUB 4080:REM DIBUJA LA PANTALLA
1840 PRINT CHR$(30);
1850 GOSUB 1910 :REM LEE EL TECLADO
1860 GOSUB 2060 :REM MUEVE EL COME-COME
1870 GOSUB 2420 :REM ENEMIGO NUMERO 1
1880 GOSUB 2650 :REM ENEMIGO NUMERO 2
1890 IF PT<=0 THEN GOSUB 3610:LET PA=PA+1:GOTO 1830
1900 GOTO 1850

```

```

1910 REM
1920 REM *****
1930 REM * CONTROL DEL TECLADO *
1940 REM *****
1950 REM
1960 GET I$:IF I$="" THEN LET I$=Z$
1970 LET Z$=I$
1980 LET II=ABS(-(I$="O")-2*(I$="o")-3*(I$="P")-4*(I$="p")-5*(I$="Q")-6*(I$="q")
-7*(I$="A")-8*(I$="a"))
1990 IF II=0 THEN RETURN
2000 LET X1=X0:LET Y1=Y0
2010 LET XO=X0+I(1,II)+I(1,II)*(S(Y0,X0+I(1,II)))=0)
2020 LET YO=Y0+I(2,II)+I(2,II)*(S(Y0+I(2,II),X0)=0)
2030 IF XO=40 THEN LET XO=0
2040 IF XO=-1 THEN LET XO=39
2050 RETURN
2060 REM
2070 REM *****
2080 REM * MOVIMIENTO DEL COME-COME *
2090 REM *****
2100 REM
2110 IF BB>0 THEN LET BB=BB+1:IF BB=60-(10*PA) THEN LET BB=0
2120 IF X1=X0 AND Y1=Y0 THEN LET Z$="Z":RETURN
2130 IF S(Y0,X0)=-1 THEN LET PT=PT-1:LET S(Y0,X0)=2:LET PU=PU+10:POKE 211,12:POKE
E 214,1:PRINT PU
2140 IF S(Y0,X0)=1 THEN LET PT=PT-1:LET S(Y0,X0)=2:LET PU=PU+50:POKE 211,12:POKE
214,1:PRINT PU:LET BB=1
2150 IF PU>=HI THEN POKE 211,31:POKE 214,1:PRINT PU:LET HI=PU
2160 POKE 211,X1:POKE 214,Y1:PRINT " "
2170 LET S(Y1,X1)=2
2180 IF SW=1 THEN POKE 211,X0:POKE 211,Y0:PRINT CHR$(81):LET SW=0:LET S(Y0,X0)=9
9:RETURN
2190 ON II GOSUB 2220,2220,2270,2270,2320,2320,2370,2370
2200 LET SW=1:LET S(Y0,X0)=99
2210 RETURN
2220 REM
2230 REM *** MOVIMIENTO A LA IZQUIERDA ***
2240 REM
2250 POKE 211,X0:POKE 214,Y0:PRINT CHR$(87)
2260 RETURN
2270 REM
2280 REM *** MOVIMIENTO A LA DERECHA ***
2290 REM
2300 POKE 211,X0:POKE 214,Y0:PRINT CHR$(87)
2310 RETURN
2320 REM
2330 REM *** MOVIMIENTO A ARRIBA ***
2340 REM
2350 POKE 211,X0:POKE 214,Y0:PRINT CHR$(87)
2360 RETURN
2370 REM
2380 REM *** MOVIMIENTO A ABAJO ***
2390 REM
2400 POKE 211,X0:POKE 214,Y0:PRINT CHR$(87)
2410 RETURN
2420 REM
2430 REM *****
2440 REM * MOVIMIENTO DEL FANTASMA-1 *
2450 REM *****
2460 REM
2470 REM *** CALCULO DE LA DIRECCION ***
2480 REM
2490 LET SO=0:LET MM=MM+1
2500 IF BB>0 AND S1=1 THEN LET S1=0:RETURN
2510 LET S1=1
2520 LET X3=X2:LET Y3=Y2
2530 LET SX=SGN(X1-X2)

```

## MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

2540 LET SY=SGN(Y1-Y2)
2550 LET X2=X2+SX+SX*(S(Y2,X2+SX)=0)
2560 LET Y2=Y2+SY+SY*(S(Y2+SY,X2)=0)
2570 REM
2580 REM *** MOVIMIENTO ***
2590 REM
2600 POKE 211,X3:POKE 214,Y3:PRINT M$(1,M1+1)
2610 LET M1=S(Y2,X2)
2620 IF M1=99 THEN GOTO 2900
2630 IF E1=1 THEN LET E1=0:POKE 211,X2:POKE 214,Y2:PRINT CHR$(120):RETURN
2640 LET E1=1:POKE 211,X2:POKE 214,Y2:PRINT CHR$(120):RETURN
2650 REM
2660 REM *****
2670 REM * MOVIMIENTO DEL FANTASMA-2 *
2680 REM *****
2690 REM
2700 REM *** CALCULO DE LA DIRECCION ***
2710 REM
2720 LET SO=0
2730 IF BB>0 AND LET S2=1 THEN LET S2=0:RETURN
2740 LET S2=1
2750 IF X3=X4 AND Y3=Y4 THEN LET MM=9
2760 IF MM<15 THEN POKE 211,X4:POKE 214,Y4:PRINT CHR$(115):RETURN
2770 LET X5=X4:LET Y5=Y4
2780 LET SX=SGN(X1-X4)
2790 LET SY=SGN(Y1-Y4)
2800 LET X4=X4+SX+SX*(S(Y4,X4+SX)=0)
2810 LET Y4=Y4+SY+SY*(S(Y4+SY,X4)=0)
2820 REM
2830 REM *** MOVIMIENTO ***
2840 REM
2850 POKE 211,X5:POKE 214,Y5:PRINT M$(2,M2+1)
2860 LET M2=S(Y4,X4)
2870 IF M2=99 THEN GOTO 2900
2880 IF E2=1 THEN LET E2=0:POKE 211,X4:POKE 214,Y4:PRINT CHR$(115):RETURN
2890 LET E2=1:POKE 211,X4:POKE 214,Y4:PRINT CHR$(150):RETURN
2900 REM
2910 REM *****
2920 REM * PERDIDA DE UNA VIDA *
2930 REM *****
2940 REM
2950 POKE 211,X0:POKE 214,Y0:PRINT " "
2960 FOR I=1 TO 10
2970 POKE 211,X0:POKE 214,Y0:PRINT CHR$(113)
2980 FOR K=1 TO 100
2990 NEXT K
3000 POKE 211,X0:POKE 214,Y0:PRINT " "
3010 FOR K=1 TO 100
3020 NEXT K
3030 NEXT I
3040 POKE 211,X0:POKE 214,Y0:PRINT " "
3050 POKE 211,X2:POKE 214,Y2:PRINT " "
3060 POKE 211,X4:POKE 214,Y4:PRINT " "
3070 LET NV=NV-1
3080 IF NV=0 THEN GOTO 3330
3090 LET Y0=15
3100 LET X0=18
3110 LET Y1=Y0
3120 LET X1=X0
3130 LET Z$="Z"
3140 LET BB=0
3150 LET X2=18
3160 LET Y2=10
3170 LET X3=X2
3180 LET Y3=Y2
3190 LET X4=20
3200 LET Y4=10

```

```

3210 LET X5=X4
3220 LET Y5=Y4
3230 LET M1=S(Y2,X2)
3240 LET M2=S(Y4,X4)
3250 POKE 211,9:POKE 214,23:PRINT " ";
3260 FOR I=1 TO NV-1
3270 POKE 211,8+I:POKE 214,23:PRINT CHR$(119)
3280 NEXT I
3290 POKE 211,X0:POKE 214,Y0:PRINT CHR$(119)
3300 GOSUB 3490
3310 RETURN
3320 RETURN
3330 REM
3340 REM *****
3350 REM * GAME OVER *
3360 REM *****
3370 REM
3380 FOR I=1 TO 5
3390 FOR J=1 TO 3
3400 POKE 211,17:POKE 214,12:PRINT " GAME "
3410 POKE 211,17:POKE 214,13:PRINT " OVER "
3420 FOR K=1 TO 140:NEXT K
3430 NEXT J
3440 POKE 211,17:POKE 214,12:PRINT " "
3450 POKE 211,17:POKE 214,13:PRINT " "
3460 FOR K=1 TO 140:NEXT K
3470 NEXT I
3480 GOTO 1800
3490 REM
3500 REM *****
3510 REM * ESTAS LISTO *
3520 REM *****
3530 REM
3540 POKE 211,17:POKE 214,12:PRINT "(ESTAS"
3550 POKE 211,17:POKE 214,13:PRINT "LISTO?"
3560 FOR J=1 TO 1000
3570 NEXT J
3580 POKE 211,17:POKE 214,12:PRINT " "
3590 POKE 211,17:POKE 214,13:PRINT " "
3600 RETURN
3610 REM
3620 REM *****
3630 REM * TERMINO DE UNA PANTALLA *
3640 REM *****
3650 REM
3660 FOR I=1 TO 10
3670 POKE 53280!,I
3680 FOR J=1 TO 100
3690 NEXT J
3700 POKE 53280!,0
3710 FOR J=1 TO 100
3720 NEXT J
3730 NEXT I
3740 PRINT CHR$(147);
3750 PRINT CHR$(31);
3760 PRINT "MUY BIEN"
3770 PRINT
3780 PRINT TAB(10);"TERMINASTE LA PANTALLA"
3790 FOR I=2 TO 40
3800 POKE 211,I-1:POKE 214,9
3810 PRINT " ."
3820 NEXT I
3830 FOR I=1 TO 10
3840 POKE 211,0:POKE 211,9
3850 PRINT CHR$(113)
3860 FOR J=1 TO 100

```

## MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

3870 NEXT J
3880 POKE 211,0:POKE 214,9
3890 PRINT " "
3900 FOR J=1 TO 100
3910 NEXT J
3920 NEXT I
3930 POKE 211,0:POKE 214,9:
3940 PRINT CHR$(113)
3950 FOR I=2 TO 40
3960 POKE 211,I-1:POKE 214,9:PRINT " ";CHR$(113)
3970 FOR J=1 TO 100:NEXT J
3980 POKE 211,I-1:POKE 214,9:PRINT " ";CHR$(119)
3990 FOR J=1 TO 100:NEXT J
4000 NEXT I
4010 PRINT
4020 PRINT " HAS CONSEGUIDO ..."
4030 PRINT
4040 PRINT PA*600;"PUNTOS DE BONUS"
4050 LET PU=PU+PA*600
4060 GOSUB 5310
4070 RETURN
4080 REM
4090 REM *****
4100 REM * DIBUJO DE LA PANTALLA *
4110 REM *****
4120 REM
4130 PRINT CHR$(147);
4140 RESTORE 6160
4150 FOR I=0 TO 24
4160 IF I=24 THEN POKE 211,0:POKE 214,24
4170 READ A$
4180 FOR J=1 TO LEN(A$)
4190 LET S(I,J)=0
4200 LET B$=MID$(A$,J,1)
4210 IF B$="." OR B$="*" THEN PRINT B$;:LET S(I,J)=(B$=",")-(B$="*"):GOTO
4220 IF B$>"9" THEN GOTO 4240
4230 PRINT CHR$(N(VAL(B$)));:GOTO 4270
4240 IF B$="A" THEN PRINT CHR$(177);
4250 IF B$="B" THEN PRINT CHR$(178);
4260 IF B$="C" THEN PRINT CHR$(162);
4270 NEXT J
4280 NEXT I
4290 POKE 211,3:POKE 214,1:PRINT "PUNTOS = ";PU
4300 POKE 211,26:POKE 214,1:PRINT "HI = ";HI
4310 POKE 211,1:POKE 214,23:PRINT "VIDAS = ";
4320 POKE 211,27:POKE 214,23:PRINT "PANTALLA = ";PA;
4330 FOR I=1 TO NV-1
4340 POKE 211,8+I:POKE 214,23:PRINT CHR$(113)
4350 NEXT I
4360 LET Y0=15
4370 LET X0=19
4380 LET Y1=Y0
4390 LET X1=X0
4400 LET Z$="Z"
4410 LET S(13,41)=2
4420 LET S(13,0)=2
4430 LET BB=0
4440 LET MM=0
4450 LET M2=-1.
4460 LET PT=287
4470 LET X2=18
4480 LET Y2=10
4490 LET X3=X2
4500 LET Y3=Y2
4510 LET X4=20

```

```

4520 LET Y4=10
4530 LET X5=X4
4540 LET Y5=Y4
4550 LET M1=-1
4560 POKE 211,X0:POKE 214,Y0
4570 PRINT CHR$(113)
4580 GOSUB 3490
4590 RETURN
4600 REM
4610 REM *****
4620 REM * TABLA DE RECORDS *
4630 REM *****
4640 REM
4650 PRINT CHR$(147)
4660 PRINT TAB(13);"TABLA DE RECORDS"
4670 PRINT TAB(12);"=====
4680 PRINT TAB(6);CHR$(176);:FOR I=1 TO 28:PRINT CHR$(99);:NEXT I:PRINT CHR$(174
)
4690 PRINT TAB(6);CHR$(98);:FOR I=1 TO 28:PRINT " ";:NEXT I:PRINT CHR$(98)
4700 PRINT TAB(6);CHR$(171);:FOR I=1 TO 28:PRINT CHR$(99);:NEXT I:PRINT CHR$(179
)
4710 FOR I=1 TO 9
4720 PRINT TAB(6);CHR$(98);:FOR I=1 TO 28:PRINT " ";:NEXT I:PRINT CHR$(98)
4730 NEXT I
4740 PRINT TAB(6);CHR$(173);:FOR I=1 TO 28:PRINT CHR$(99);:NEXT I:PRINT CHR$(125
)
4750 NP=0
4760 FOR I=1 TO 10
4770 IF PU>P(I) THEN LET NP=1
4780 NEXT I
4790 IF NP=1 THEN LET P(10)=PU:LET N$(10)=CHR$(255)
4800 FOR I=1 TO 10
4810 FOR J=1 TO I
4820 IF P(I)>P(J) THEN LET A$=N$(I):LET A=P(I):LET N$(I)=N$(J):LET P(I)=P(
J):LET N$(J)=A$:LET P(J)=A
4830 NEXT J
4840 NEXT I
4850 IF NP=0 THEN GOTO 4890
4860 FOR I=1 TO 10
4870 IF N$(I)=CHR$(255) THEN LET N$(I)="-----":LET NP=I
4880 NEXT I
4890 POKE 211,6:POKE 214,3:PRINT N$(1);TAB(23);"... ";P(1)
4900 FOR I=2 TO 10
4910 POKE 211,7:POKE 214,4+I
4920 PRINT N$(I);TAB(23);"... ";P(I)
4930 NEXT I
4940 IF NP=0 THEN GOTO 5120
4950 POKE 211,14:POKE 214,15:PRINT "FELICIDADES."
4960 PRINT
4970 PRINT TAB(3);"TU PUNTUACION ESTA ENTRE LAS MEJORES"
4980 PRINT
4990 PRINT TAB(12);"ESCRIBE TU NOMBRE."
5000 LET N$(NP)="" :LET X=6:LET LO=0:LET D$=""
5010 IF NP=1 THEN LET Y=3 ELSE LET Y=3+NP
5020 POKE 211,X:POKE 214,Y
5030 PRINT " ";CHR$(157);
5040 GET A$:IF A$="" THEN GOTO 5040
5050 IF A$=CHR$(20) AND LO>0 THEN LET X=X-1:LET LO=LO-1:LET D$=LEFT$(D$,LO):PRIN
T CHR$(157);"_" :CHR$(157);CHR$(157);
5060 IF A$=" " THEN GOTO 5090
5070 IF A$=CHR$(13) THEN GOTO 5100
5080 IF A$<"0" OR A$>"z" THEN GOTO 5040
5090 PRINT A$;"_";CHR$(157);:LET LO=LO+1:LET D$=D$+A$:IF LO<>15 THEN GOTO 5040
5100 FOR I=1 TO 16-LO:PRINT " ";:NEXT I
5110 LET N$(NP)=D$
5120 LET HI=P(1)

```

## MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

5130 POKE 211,0:POKE 214,16:FOR I=1 TO 240:PRINT " ";:NEXT I
5140 LET PU=0:LET NV=3
5150 IF NP<>0 THEN LET NP=0:GOSUB 5410:GOTO 4800
5160 RETURN
5170 REM
5180 REM *****
5190 REM * INSTRUCCIONES *
5200 REM *****
5210 REM
5220 POKE 211,3:POKE 214,16
5230 PRINT "COMETE TODOS LOS PUNTOS QUE PUEDAS"
5240 PRINT
5250 PRINT " ^"
5260 PRINT " Q USA ESTAS TECLAS"
5270 PRINT "< O P >"
5280 PRINT " A PARA MOVER EL QUESITO."
5290 PRINT " v"
5300 RETURN
5310 REM
5320 REM *****
5330 REM * PULSA UNA TECLA *
5340 REM *****
5350 REM
5360 POKE 211,12:POKE 214,24
5370 PRINT "PULSA UNA TECLA"
5380 GET A$
5390 IF A$="" THEN GOTO 5380
5400 RETURN
5410 REM
5420 REM *****
5430 REM * PRESENTACION *
5440 REM *****
5450 REM
5460 PRINT CHR$(147)
5470 RESTORE 6480
5480 FOR I=2 TO 16
5490 READ A$
5500 FOR J=1 TO LEN(A$)
5510 IF MID$(A$,J,1)="O" THEN GOTO 5540
5520 POKE 211,6+J:POKE 214,I
5530 PRINT CHR$(113)
5540 NEXT J
5550 NEXT I
5560 POKE 211,5:POKE 214,0:PRINT "(c) Ed. Siglo Cultural, 1987"
5570 FOR I=1 TO 14
5580 POKE 211,I-1:POKE 214,19:PRINT " ."
5590 FOR J=1 TO 100:NEXT J
5600 NEXT I
5610 POKE 211,16:POKE 214,19:PRINT "= 10 PUNTOS"
5620 FOR I=1 TO 13
5630 POKE 211,I-1:POKE 214,21:PRINT " *"
5640 FOR J=1 TO 100:NEXT J
5650 NEXT I
5660 POKE 211,16:POKE 214,21:PRINT "= 50 PUNTOS"
5670 FOR J=1 TO 1000:NEXT J
5680 FOR I=1 TO 15
5690 POKE 211,I-1:POKE 214,19:PRINT CHR$(113)
5700 FOR J=1 TO 100:NEXT J
5710 POKE 211,I-1:POKE 214,19:PRINT CHR$(119)
5720 FOR J=1 TO 100:NEXT J
5730 POKE 211,I-1:POKE 214,19:PRINT " "
5740 NEXT I
5750 POKE 211,14:POKE 214,19:PRINT CHR$(119)
5760 POKE 211,16:POKE 214,19:PRINT "= TU MISMO "
5770 FOR I=1 TO 15
5780 POKE 211,I-1:POKE 214,21:PRINT CHR$(115)

```

```

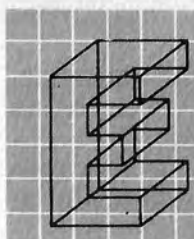
5790 FOR J=1 TO 100:NEXT J
5800 POKE 211,I-1:POKE 24,21:PRINT CHR$(120)
5810 FOR J=1 TO 100:NEXT J
5820 POKE 211,I-1:POKE 214,21:PRINT " "
5830 NEXT I
5840 POKE 211,14:POKE 214,21:PRINT CHR$(115)
5850 POKE 211,16:POKE 211,21:PRINT "= TU ENEMIGO"
5860 FOR I=1 TO 1000
5870 NEXT I
5880 RETURN
5890 REM
5900 REM *****
5910 REM * DATAS CON LOS CARACTERES *
5920 REM *****
5930 REM
5940 DATA 32,176,174,173,125,99,98,123,171,179
5950 REM
5960 REM *****
5970 REM * DATAS CON LOS RECORDS *
5980 REM *****
5990 REM
6000 DATA "ANTONIO",34980
6010 DATA "EL COCO",31720
6020 DATA "EL AUTOR",24030
6030 DATA "INDIANA",23300
6040 DATA "JOSELITO",11210
6050 DATA "SUSYN",11120
6060 DATA "RAMIRO",8430
6070 DATA "DONALD",7540
6080 DATA "SHE",3400
6090 DATA "HE",3210
6100 REM
6110 REM *****
6120 REM * DATAS CON LAS DIRECCIONES *
6130 REM *****
6140 REM
6150 DATA -1,-1,1,1,0,0,0,0,0,0,0,-1,-1,1,1
6160 REM
6170 REM *****
6180 REM * DATAS CON LA DEFINICION DE LA PANTALLA *
6190 REM *****
6200 REM
6210 DATA "001555555555555555200000015555555555200"
6220 DATA "006000000000000000006000000000000000600"
6230 DATA "15A5555555555555B5A5555B5A55555555555A52"
6240 DATA "6*.....35555554.....*6"
6250 DATA "6.15552.1555552.....1555552.15552.6"
6260 DATA "6.60006.600000352.1552.154000006.60006.6"
6270 DATA "6.35554.355555554.6006.355555554.35554.6"
6280 DATA "6.....6006.....6"
6290 DATA "6555552.155552.1554003552.155552.155559"
6300 DATA "6155526.601554.355555554.355206.6155526"
6310 DATA "6355546.606.....606.6355546"
6320 DATA "6355554.354.152.1CCCCC2.152.354.3555554"
6330 DATA ".....606.60000006.606....."
6340 DATA "15555B2.152.606.60000006.606.152.1B55552"
6350 DATA "6...34.354.354.35555554.354.354.34...6"
6360 DATA "6.12.....12.6"
6370 DATA "6.63552.155555555555555555552.15546.6"
6380 DATA "6.35526.60001555555555555520006.61554.6"
6390 DATA "6...34.35554.....35554.34...6"
6400 DATA "6.12.....15B55555B52.....12.6"
6410 DATA "6.34.15555552.354.....354.15555552.34.6"
6420 DATA "6*...35555206.....1552.....60155554...*6"
6430 DATA "655555555A575555A55A5555BA5A555555559"
6440 DATA "600000000006000000000006000000000006"

```

## MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
6450 DATA "35555555555540000000000000003555555555554"
6460 REM
6470 REM *****
6480 REM * DATAS CON LA PALABRA PAC-MAN *
6490 REM *****
6500 REM
6510 DATA "1111110000110000011110"
6520 DATA "0110011001111000110011"
6530 DATA "0110Q11011001101100000"
6540 DATA "0111100011001101100000"
6550 DATA "0110000011111101100000"
6560 DATA "0110000011001100110011"
6570 DATA "1111000011001100011110"
6580 DATA ""
6590 DATA "11000110000110001100011"
6600 DATA "11101110001111001110011"
6610 DATA "11111110011001101111011"
6620 DATA "11111110011001101101111"
6630 DATA "11010110011111101100111"
6640 DATA "11000110011001101100011"
6650 DATA "11000110011001101100011"
```

# TRUCOS Y RUTINAS BASICAS



N este nuevo tomo vamos a ver un programa Monitor de memoria con el que podremos testear, listar, mover, copiar, y, en definitiva, hacer lo que queramos con la memoria y con su contenido.

El programa se realizó inicialmente en un IBM bajo GWBASIC. Para que funcione en otros ordenadores se proponen los siguientes cambios:

## AMSTRAD:

```

1160 CLS
1170 MODE 1
1190 REM
1200 LET CH=0
1310 LOCATE 23,2
1450 IF K$="0" THEN LOCATE 1,23: INPUT
"VALOR A POKEAR = "; P: IF P>255 OR P>0 THEN
GOTO 1450
1460 IF K$="0" THEN POKE MP, P: LOCATE
1,23: PRINT SPACE$(40): GOTO 1300
1790 LOCATE 1,11: PRINT ">"
1800 IF MO= 1 THEN LOCATE 16,11: PRINT ">"
1820 IF MO= 3 THEN LOCATE 10,11: PRINT ">"
1890 LOCATE 1,23: INPUT "DESDE LA DIR. ";
I: LOCATE 1,23: PRINT SPACE$(40): 1910 LOCATE
1,23: INPUT "HASTA LA DIR. "; F: LOCATE 1,23:
PRINT SPACE$(40)
1960 LET CH=8
2000 LET CH=0
2260 LOCATE 1,23
2280 LOCATE 1,23: PRINT SPACE$(40)
2320 LOCATE 1,22
2340 LOCATE 1,23
2360 LOCATE 1,23: PRINT SPACE$(40)
2400 LOCATE 1,23
2510 LOCATE 1,23
2620 LOCATE 1,23
2640 LOCATE 1,23: PRINT SPACE$(40)
2660 LOCATE 1,23

```

```

2790 REM
2800 REM
2810 LOAD D$,D
2880 LOCATE 1,23
2890 REM
2900 REM
2910 REM
2920 LOCATE 1,23: PRINT SPACE$(40)
2930 LOCATE 1,23
2960 LOCATE 1,23: PRINT SPACE$(40)
2970 LOCATE 1,23
3000 LOCATE 1,23: PRINT SPACE$(40)
3080 LOCATE 1,23: PRINT SPACE$(40)
3090 LOCATE 1,23
3120 REM
3130 REM
3140 SAVE D$,B,D,D+L,D
3210 LOCATE 1,23: PRINT SPACE$(40)
3220 LOCATE 1,23
3250 LOCATE 1,23: PRINT SPACE$(40)
3330 LOCATE 19,6: PRINT "1. BASE 2."
3340 LOCATE 19,8: PRINT "2. BASE 10."
3350 LOCATE 19,10: PRINT "3. BASE 16."
3480 LOCATE 1,23: PRINT SPACE$(40)
3490 LOCATE 1,23
3520 LOCATE 1,23: PRINT SPACE$(40)
3530 LOCATE 1,23
3560 LOCATE 1,23: PRINT SPACE$(40)
3570 LOCATE 1,23
3980 PEN 1: PRINT "D";: PEN 2: PRINT "UMP.
CAMBIA MODO"
3990 PEN 1: PRINT "B";: PEN 2: PRINT "USCA
BYTES EN MEMORIA"
4000 PEN 1: PRINT "A";: PEN 2: PRINT "YUDA.
PONE ESTAS INSTRUCCIONES"
4010 PEN 1: PRINT "J";: PEN 2: PRINT "UMP.
EJECUTA CODIGO MAQUINA"
4020 PEN 1: PRINT "L";: PEN 2: PRINT "OAD.
CARGA BYTES"
4030 PEN 1: PRINT "M";: PEN 2: PRINT
"EMORY. AJUSTA PUNTERO"
4040 PEN 1: PRINT "P";: PEN 2: PRINT "RIN-
TER. PASA POR LA IMPRESORA"
4050 PEN 1: PRINT "S";: PEN 2: PRINT "AVE.
GRABA BYTES"

```

## TRUCOS Y RUTINAS BASICAS

4060 PEN 1: PRINT "T"; PEN 2: PRINT "RAS-  
LADA BYTES"

" =4070 PEN 1: PRINT "X"; PEN 2: PRINT "=  
CAMBIA DE BASE"

4080 PEN 1: PRINT "2"; PEN 2: PRINT " = AU-  
MENTA EL PUNTERO EN 10"

4090 PEN 1: PRINT "1"; PEN 2: PRINT " = AU-  
MENTA EL PUNTERO EN UNO "

4100 PEN 1: PRINT "9"; PEN 2: PRINT " = DIS-  
MINUYE EL PUNTERO EN UNO"

4110 PEN 1: PRINT "8"; PEN 2: PRINT " = DIS-  
MINUYE EL PUNTERO EN 0 O EN 10"

4120 PEN 1: PRINT "0"; PEN 2: PRINT " = PO-  
KEA EN EL PUNTERO "

4420 LOCATE 1,23: PRINT "CONTINUAMOS?  
(S/N)"

4460 LOCATE 1,23: PRINT SPACE\$(40)

### MSX:

Hay que realizar las mismas que para el AMSTRAD, excepto las siguientes líneas:

1180 OPNE "CRT: " FOR OUTPUT AS # 1  
OPEN

1190 OPEN "LPT:" FOR OUTPUT AS# 2

1200 LET CH=1

1960 LET CH=2

2000 LET CH=1

También hay que quitar todas las sen-  
tencias PEN comprendidas entre las líneas  
3980 y 4120.



Fig. 1. Comandos del programa.

Con este programa podremos moldear la memoria según nuestros deseos. Su uso es muy sencillo y sólo requiere un poco de práctica.

Al usarlo hay que tener cuidado de no meternos en zonas prohibidas de la memoria, como pueden ser las variables del sistema, la ROM (haciendo llamadas a tontas y a locas), etcétera.

Los comandos de que dispone el programa son los siguientes:

D - DUMP. Cambia el modo en que vemos el contenido de la memoria. Inicialmente, lo veremos tal y como aparece en la figura 2. Si pulsamos la D, pasaremos al modo DUMP y lo veremos como nos muestra la figura 3. Volviendo a pulsar la D, volveremos al modo anterior.

B - BUSCAR. Nos permite buscar hasta cinco bytes consecutivos, y que nosotros conocemos, en la memoria.

A - AYUDA. Nos imprime una lista con todos los comandos disponibles.

J - JUMP. Realiza un salto a la dirección de memoria que nosotros le indiquemos.

L - LOAD. Nos permite cargar bytes desde disco o cinta.

M - MEMORY. Con él podremos cambiar el puntero que apunta a la dirección de memoria que estamos visualizando.

P - Imprime el contenido de una zona de memoria en la impresora. Se puede utilizar en ambos modos de DUMP.

S - SAVE. Nos permite grabar el contenido de una porción de la memoria en disco o cinta.

T - TRASLADA. Nos permite copiar una serie de bytes que se encuentran en una zona de la memoria a otra zona distinta.

X - BASE. Nos permite definir la base en que queremos que aparezca el DUMP de memoria. Esta puede ser base 2, 10 ó 16.

1 - Aumenta el puntero de la memoria en uno.

2 - Disminuye el puntero de memoria en uno.

8 - Disminuye el puntero de memoria en 10.

9 - Aumenta el puntero de memoria en 10.

0 - Nos permite introducir un valor entre 0 y 255 en la posición actual del puntero.



Fig. 2. DUMP de memoria tipo A.

```

1000 REM *****
1010 REM * MONITOR DE MEMORIA *
1020 REM *****
1030 REM
1040 REM *****
1050 REM *
1060 REM * (c) Ediciones Siglo Cultural *
1070 REM *
1080 REM * (c) 1977 *
1090 REM *
1100 REM *****
1110 REM
1120 REM *****
1130 REM * INICIAL ZACION *
1140 REM *****
1150 REM
1160 SCREEN 0
1170 WIDTH 40
1180 OPEN "SCRN:" FOR OUTPUT AS #1
1190 OPEN "LPT1:" FOR OUTPUT AS #2
1200 LET CH=1
1210 LET BA=10
1220 LET MP=0
1230 LET MO=2
1240 LET SW=1
1250 REM
1260 REM *****
1270 REM * PROGRAMA PRINCIPAL *
1280 REM *****
1290 REM
1300 CLS
1310 LOCATE 2,23
1320 PRINT "ESTOY EN BASE ";BA
1330 GOSUB 1590
1340 LET K$=INKEY$
1350 IF K$="" THEN GOTO 1340
1360 IF K$="1" THEN LET MP=MP-1
1370 IF MP<0 THEN LET MP=65535!
1380 IF K$="2" THEN LET MP=MP+1
1390 IF MP>65535! THEN LET MP=0
1400 IF K$="8" THEN LET MP=MP-10
1410 IF MP<0 THEN LET MP=65535!+MP
1420 IF K$="9" THEN LET MP=MP+10
1430 IF MP>65535! THEN LET MP=MP-65535!
1440 IF K$="0" AND K$<"." THEN GOSUB 1590:GOTO 1340
1450 IF K$="0" THEN LOCATE 23,1:INPUT "VALOR A POKEAR = ";P:IF P>255 OR P<0 THEN
GOTO 1450
1460 IF K$="0" THEN POKE MP,P:LOCATE 23,1:PRINT SPACE$(40):GOTO 1300
1470 IF K$="A" THEN GOSUB 3880:GOTO 1300
1480 IF K$="B" THEN GOSUB 2210:GOTO 1340
1490 IF K$="N" THEN GOSUB 2400:GOTO 1340
1500 IF K$="J" THEN GOSUB 2570:GOTO 1300
1510 IF K$="L" THEN GOSUB 2730:GOTO 1300
1520 IF K$="S" THEN GOSUB 3020:GOTO 1300
1530 IF K$="M" THEN GOSUB 3180:GOTO 1300
1540 IF K$="X" THEN GOSUB 3280:GOTO 1300
1550 IF K$="P" THEN GOSUB 1840:GOTO 1300
1560 IF K$="T" THEN GOSUB 3430:GOTO 1300
1570 IF K$="D" THEN LET SW=-SW:CLS:GOTO 1300
1580 GOTO 1340
1590 REM
1600 REM *****
1610 REM * DISPLAY DE LA MEMORIA *

```

## TRUCOS Y RUTINAS BASICAS

```

1820 REM *****
1830 REM
1840 IF SW=-1 THEN GOTO 4190
1850 LOCATE 1,1
1860 FOR A=MP-10 TO MP+11
1870 LET B=A
1880 IF A>85535! THEN LET A=A-85536!
1890 IF A<0 THEN LET A=85536!+A
1900 LET N$=MID$(STR$(A),2,LEN(STR$(A))-1)
1910 LET P$=MID$(STR$(PEEK(A)),2,LEN(STR$(PEEK(A))-1))
1920 IF MO=1 THEN LET NU=VAL(P$):GOSUB 3840
1930 IF LEN(N$)<5 THEN LET N$="0"+N$:GOTO 1730
1940 IF MO=2 THEN IF LEN(P$)<3 THEN LET P$="0"+P$:GOTO 1740
1950 IF MO=3 THEN LET NU=VAL(P$):GOSUB 2030
1960 PRINT#CH," ";N$," ";P$
1970 LET A=B
1980 NEXT A
1990 LOCATE 11,1:PRINT ">"
2000 IF MO=1 THEN LOCATE 11,16:PRINT "<"
2010 IF MO=2 THEN LOCATE 11,11:PRINT "<"
2020 IF MO=3 THEN LOCATE 11,10:PRINT "<"
2030 RETURN
2040 REM
2050 REM *****
2060 REM * SALIDA POR IMPRESORA *
2070 REM *****
2080 REM
2090 LOCATE 23,1:INPUT "DESDE LA DIR. ";I:LOCATE 23,1:PRINT SPACE$(40)
2100 IF I<0 THEN GOTO 1890
2110 LOCATE 23,1:INPUT "HASTA LA DIR. ";F:LOCATE 23,1:PRINT SPACE$(40)
2120 IF F<I OR F>85535! THEN GOTO 1910
2130 LET A=MP
2140 FOR M=I TO F STEP 22
2150 LET MP=M
2160 LET CH=2
2170 GOSUB 1590
2180 NEXT M
2190 LET MP=A
2200 LET CH=1
2210 GOSUB 1590
2220 RETURN
2230 REM
2240 REM *****
2250 REM * PASO DE DECIMAL A HEXADECIMAL *
2260 REM *****
2270 REM
2280 LET P$=""
2290 FOR C=1 TO 0 STEP -1
2300 LET N=INT(NU/16^C)
2310 LET NU=NU-N*16^C
2320 IF N<10 THEN LET P$=P$+MID$(STR$(N),2,1)
2330 IF N=10 THEN LET P$=P$+"A"
2340 IF N=11 THEN LET P$=P$+"B"
2350 IF N=12 THEN LET P$=P$+"C"
2360 IF N=13 THEN LET P$=P$+"D"
2370 IF N=14 THEN LET P$=P$+"E"
2380 IF N=15 THEN LET P$=P$+"F"
2390 NEXT C
2400 RETURN
2410 REM
2420 REM *****
2430 REM * BUSQUEDA DE BYTES *
2440 REM *****

```

```

2250 REM
2260 LOCATE 23,1
2270 INPUT "CUANTOS BUSCO? (1-5) ";B
2280 LOCATE 23,1:PRINT SPACE$(40)
2290 IF B>5 OR B<1 THEN GOTO 2260
2300 FOR A=1 TO B
2310 CLS
2320 LOCATE 22,1
2330 PRINT "DAME EL NUMERO ";A
2340 LOCATE 23,1
2350 INPUT "NUMERO > ";C(A)
2360 LOCATE 23,1:PRINT SPACE$(40)
2370 IF C>255 OR C<0 THEN GOTO 2350
2380 NEXT A
2390 CLS
2400 LOCATE 23,1
2410 PRINT "ESPERE UN MOMENTO. BUSCANDO"
2420 FOR I=0 TO 65535!
2430 LET NN=0
2440 FOR J=0 TO B-1
2450 IF PEEK(I+J)=C(J+1) THEN NN=NN+1
2460 NEXT J
2470 IF NN=B THEN LET A=I:GOTO 2490
2480 NEXT I
2490 LET MP=A
2500 CLS
2510 LOCATE 23,1
2520 PRINT "ENCONTRADO. PULSE UNA TECLA"
2530 LET A$=INKEY$
2540 IF A$="" THEN GOTO 2530
2550 GOSUB 1590
2560 RETURN
2570 REM
2580 REM *****
2590 REM * SALTO A UNA DIRECCION DE MEMORIA *
2600 REM *****
2610 REM
2620 LOCATE 23,1
2630 INPUT "A QUE DIRECCION SALTO? (0-65535) ";J
2640 LOCATE 23,1:PRINT SPACE$(40)
2650 IF J<0 OR J>65535! THEN GOTO 2620
2660 LOCATE 23,1
2670 PRINT "ESTAS SEGURO? (S/N)"
2680 LET A$=INKEY$
2690 IF A$="" THEN GOTO 2680
2700 IF A$="S" OR A$="s" THEN CALL J
2710 IF A$<>"N" AND A$<>"n" THEN GOTO 2680
2720 GOTO 4140
2730 REM
2740 REM *****
2750 REM * CARGAR BYTES (LOAD) *
2760 REM *****
2770 REM
2780 GOSUB 2830
2790 IF A$="D" OR A$="d" THEN BLOAD "CAS:"+D$:RETURN
2800 IF A$<>"C" AND A$<>"c" THEN GOTO 2780
2810 BLOAD D$
2820 RETURN
2830 REM
2840 REM *****
2850 REM * CASETE O DISCO? *
2860 REM *****
2870 REM

```

## TRUCOS Y RUTINAS BASICAS

```

2880 LOCATE 23,1
2890 PRINT "CASSETTE O DISCO (C/D) ?"
2900 LET A$=INKEY$
2910 IF A$="" THEN GOTO 2900
2920 LOCATE 23,1:PRINT SPACE$(40)
2930 LOCATE 23,1
2940 INPUT "NOMBRE? (1-8 CHRS.) ";D$
2950 IF LEN(D$)>8 OR LEN(D$)=0 THEN GOTO 2920
2960 LOCATE 23,1:PRINT SPACE$(40)
2970 LOCATE 23,1
2980 INPUT "DESPLAZAMIENTO ? ";D
2990 IF D<0 OR D>85535! THEN GOTO 2980
3000 LOCATE 23,1:PRINT SPACE$(40)
3010 RETURN
3020 REM
3030 REM *****
3040 REM * GRABAR BYTES (SAVE) *
3050 REM *****
3060 REM
3070 GOSUB 2830
3080 LOCATE 23,1:PRINT SPACE$(40)
3090 LOCATE 23,1
3100 INPUT "LONGITUD? ";L
3110 IF L>35535! OR L<0 THEN GOTO 3080
3120 IF A$="D" OR A$="d" THEN BSAVE D$,D,D+L:RETURN
3130 IF A$<>"C" AND A$<>"c" THEN GOTO 3070
3140 BSAVE "CAS:"+D$,D,D+L
3150 RETURN
3160 REM
3170 REM *****
3180 REM * CAMBIAR PUNTERO DE MEMORIA *
3190 REM *****
3200 REM
3210 LOCATE 23,1:PRINT SPACE$(40)
3220 LOCATE 23,1
3230 INPUT "MEMORY = ";M
3240 IF M>85535! OR M<0 THEN GOTO 3210
3250 LOCATE 23,1:PRINT SPACE$(40)
3260 LET MP=M
3270 RETURN
3280 REM
3290 REM *****
3300 REM * CAMBIO DE BASE *
3310 REM *****
3320 REM
3330 LOCATE 6,19:PRINT "1. BASE 2."
3340 LOCATE 8,19:PRINT "2. BASE 10."
3350 LOCATE 10,19:PRINT "3. BASE 16."
3360 LET K$=INKEY$
3370 IF K$="" THEN GOTO 3360
3380 IF K$<>"1" AND K$<>"2" AND K$<>"3" THEN GOTO 3360
3390 LET BA=-(2 AND K$="1")-(10 AND K$="2")-(16 AND K$="3")
3400 LET MO=VAL(K$)
3410 LET K$=""
3420 RETURN
3430 REM
3440 REM *****
3450 REM * MUEVE BYTES *
3460 REM *****
3470 REM
3480 LOCATE 23,1:PRINT SPACE$(40)
3490 LOCATE 23,1
3500 INPUT "DESDE? ";P

```

```

3510 IF P<0 OR P>85535! THEN GOTO 3480
3520 LOCATE 23,1:PRINT SPACE$(40)
3530 LOCATE 23,1
3540 INPUT "HASTA? ";D
3550 IF D<0 OR D>85535! THEN GOTO 3520
3560 LOCATE 23,1:PRINT SPACE$(40)
3570 LOCATE 23,1
3580 INPUT "LONGITUD? ";L
3590 IF L<0 OR L>85535! THEN GOTO 3560
3600 FOR I=0 TO L-1
3610 POKE D+L,PEEK(P+L)
3620 NEXT I
3630 RETURN
3640 REM
3650 REM *****
3660 REM * PASO DE DECIMAL A BINARIO *
3670 REM *****
3680 REM
3690 LET P$=""
3700 IF NU>=128 THEN LET P$="1":LET NU=NU-128:GOTO 3720
3710 P$="0"
3720 IF NU>=64 THEN LET P$=P$+"1":LET NU=NU-64:GOTO 3740
3730 P$=P$+"0"
3740 IF NU>=32 THEN LET P$=P$+"1":LET NU=NU-32:GOTO 3760
3750 P$=P$+"0"
3760 IF NU>=16 THEN LET P$=P$+"1":LET NU=NU-16:GOTO 3780
3770 P$=P$+"0"
3780 IF NU>=8 THEN LET P$=P$+"1":LET NU=NU-8:GOTO 3800
3790 P$=P$+"0"
3800 IF NU>=4 THEN LET P$=P$+"1":LET NU=NU-4:GOTO 3820
3810 P$=P$+"0"
3820 IF NU>=2 THEN LET P$=P$+"1":LET NU=NU-2:GOTO 3840
3830 P$=P$+"0"
3840 IF NU=1 THEN LET P$=P$+"1":LET NU=0:GOTO 3860
3850 P$=P$+"0"
3860 RETURN
3870 REM
3880 REM
3890 REM *****
3900 REM * INSTRUCCIONES *
3910 REM *****
3920 REM
3930 CLS
3940 PRINT "=====
3950 PRINT "*** I N S T R U C C I O N E S ***"
3960 PRINT "=====
3970 PRINT
3980 COLOR 0,7:PRINT "D";:COLOR 7,0:PRINT "DUM. CAMBIA MODO."
3990 COLOR 0,7:PRINT "B";:COLOR 7,0:PRINT "USCA BYTES EN MEMORIA"
4000 COLOR 0,7:PRINT "A";:COLOR 7,0:PRINT "YUDA. PONE ESTAS INSTRUCCIONES."
4010 COLOR 0,7:PRINT "J";:COLOR 7,0:PRINT "UMP. EJECUTA CODIGO MAQUINA."
4020 COLOR 0,7:PRINT "L";:COLOR 7,0:PRINT "OAD. CARGA BYTES."
4030 COLOR 0,7:PRINT "M";:COLOR 7,0:PRINT "EMORY. AJUSTA EL PUNTERO."
4040 COLOR 0,7:PRINT "P";:COLOR 7,0:PRINT "RINTER. PASA POR IMPRESORA."
4050 COLOR 0,7:PRINT "S";:COLOR 7,0:PRINT "AVE. GRABA BYTES."
4060 COLOR 0,7:PRINT "T";:COLOR 7,0:PRINT "RASLADA BYTES."
4070 COLOR 0,7:PRINT "X";:COLOR 7,0:PRINT " = CAMBIA DE BASE."
4080 COLOR 0,7:PRINT "2";:COLOR 7,0:PRINT " = AUMENTA EL PUNTERO EN 10."
4090 COLOR 0,7:PRINT "1";:COLOR 7,0:PRINT " = AUMENTA EL PUNTERO EN UNO."
4100 COLOR 0,7:PRINT "9";:COLOR 7,0:PRINT " = DISMINUYE EL PUNTERO EN UNO."
4110 COLOR 0,7:PRINT "8";:COLOR 7,0:PRINT " = DISMINUYE EL PUNTERO EN 10."
4120 COLOR 0,7:PRINT "0";:COLOR 7,0:PRINT " = POKEA EN EL PUNTERO."
4130 PRINT

```

## TRUCOS Y RUTINAS BASICAS

```

4140 PRINT "PULSA UNA TECLA"
4150 LET A$=INKEY$
4160 IF A$="" THEN GOTO 4150
4170 CLS
4180 RETURN
4190 REM
4200 REM *****
4210 REM * MODO DUMP DE MEMORIA *
4220 REM *****
4230 REM
4240 CLS
4250 FOR I=MP TO MP+150 STEP 10
4260 IF I>65535! THEN GOTO 4470
4270 LET N$=MID$(STR$(I),2,LEN(STR$(I))-1)
4280 IF LEN(N$)<5 THEN LET N$="0"+N$:GOTO 4280
4290 PRINT#CH,N$;" ";
4300 LET Q$=""
4310 FOR J=I TO I+9
4320 LET NU=PEEK(J)
4330 LET M$=CHR$(NU)
4340 IF NU<32 THEN M$="."
4350 GOSUB 2040
4360 Q$=Q$+M$
4370 PRINT#CH,P$;
4380 NEXT J
4390 PRINT#CH," ";Q$
4400 NEXT I
4410 LET MP=MP+150
4420 LOCATE 23,1:PRINT "CONTINUAMOS? (S/N)"
4430 LET A$=INKEY$
4440 IF A$="S" OR A$="s" THEN GOTO 4240
4450 IF A$<>"N" AND A$<>"n" THEN GOTO 4430
4460 LOCATE 23,1:PRINT SPACE$(40)
4470 RETURN

```

```

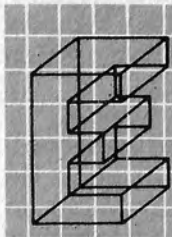
02000- 53444647484A4B4C0000 SDFGHJKL...
02010- 00000058594356424F4D ...ZACVBWN
02020- 00000000000000000000
02030- 0E1E00001D001C000E1F
02040- 17127F020605180C4101 .A.....B
02050- 42014301440145014601 B.C.D.E.F.
02060- 4701480149014A004B01 G.H.I.J.K.
02070- 4C014D024E014F015001 L.M.N.O.P.
02080- 51005201530154015501 Q.R.S.T.U.
02090- 56015701580159005A00 V.W.X.Y.Z.
02100- 00000000000000000000
02110- 0000590330000470C2E00 ...X.O.G...
02120- 020005010A0004000000 ...Lfti...
02130- 00004C50543100000000
02140- 00000000000000000000
02150- 00000000000000000000

```

CONTINUAMOS? (S/N)

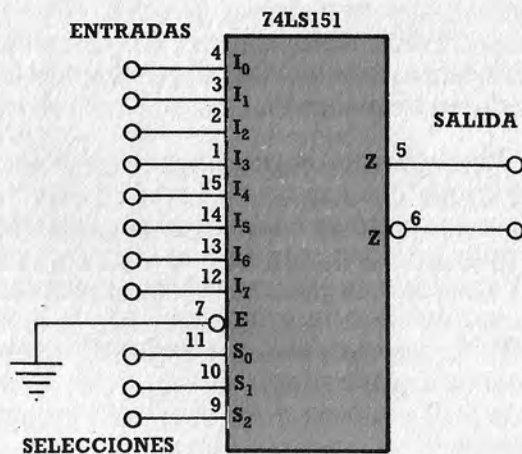
Fig. 3. DUMP de memoria tipo B.

## Central de alarmas. Segunda parte



En el tomo anterior se han descrito diversos circuitos que pueden emplearse para detección de anomalías en las instalaciones que pueden interpretarse como alarmas. Es necesario además actuar

sobre el sistema para efectuar correcciones o indicar de manera llamativa la producción de la situación de alarma. Describiremos circuitos que pueden emplearse para comunicar la alarma y programas que permitan la utilización del ordenador para la comunicación de la situación y la toma de decisiones sobre la acción a tomar.



## Lectura de las señales de alarma

Con los captadores de señales del entorno a controlar se han producido un conjunto de valores binarios que indican la existencia de una situación anómala. Mediante la tarjeta de ampliación de puertos de entrada/salida es posible acceder de forma rápida a cada una de las señales. Como en general el número de variables a captar será grande, conviene realizar algún tipo de multiplexado para reducir el número de entradas necesarias al sistema. La multiplexión de entradas, cuando éstas son de tipo binario, se realiza mediante circuitos combinatoriales de tipo multiplexor o con conexiones de tipo de tres estados, activando cada entrada en secuencia.

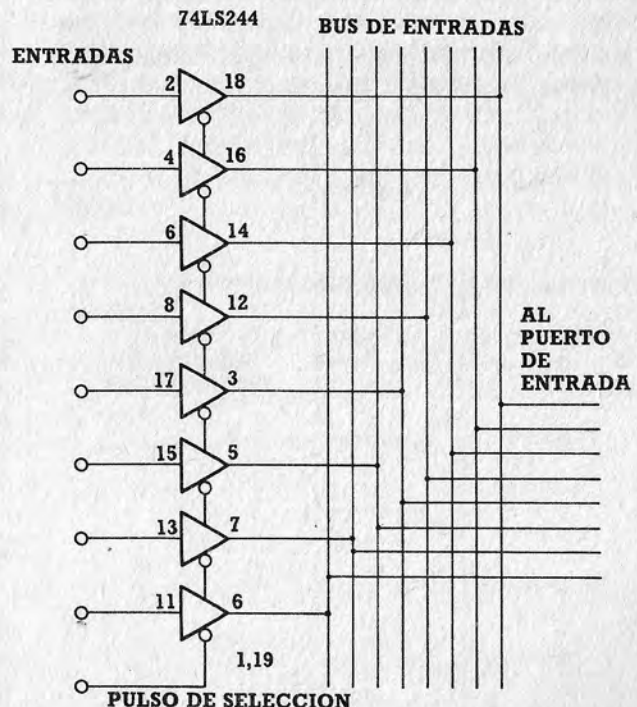


Fig. 1. Multiplexores.

## EL TALLER DEL HARDWARE

Si las entradas poseen un nivel de comparación variable es necesario dotar a cada entrada de un estado con el que efectuar la comparación. La manera de hacer variable la comparación es disponer de un registro de salida programable a razón de un bit por cada entrada de este tipo.

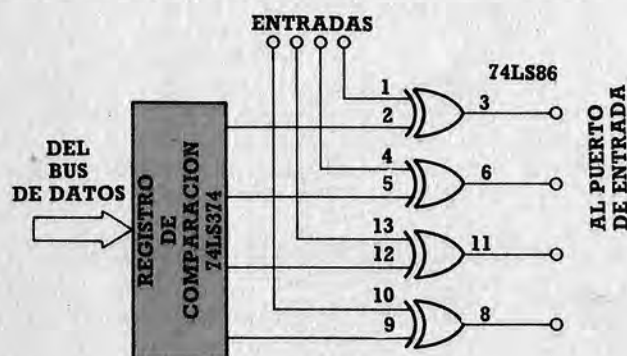


Fig. 2. Entradas de nivel de comparación programable.

Este tipo de entradas suelen conectarse de forma que al producirse algún cambio se comunique inmediatamente al ordenador, para que tome alguna acción correctiva en el menor tiempo posible. Cada entrada se conecta a una puerta de tipo OR exclusivo que da a su salida un 1 si las dos entradas son distintas y un 0 si las dos entradas son iguales. Cada entrada puede ser leída individualmente a través de un puerto. La forma de conectar al ordenador estas señales es a través de la entrada de interrupción de alguno de los tipos que el ordenador personal posea.

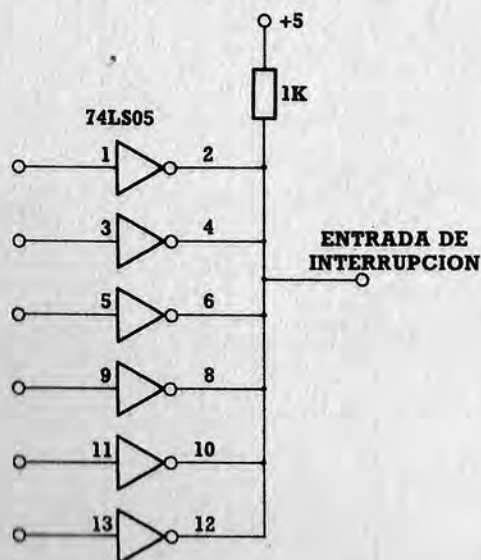


Fig. 3. Conexión a través de interrupción.

La conexión de las señales a través de entrada de interrupción requiere que el sistema esté preparado para recibirlas. La forma más normal de utilizar esta posibilidad de interrupción es a través de alguna sentencia del lenguaje que permita el cambio del flujo de la ejecución de instrucciones cada vez que se produzca un fenómeno, asincrónicamente con el programa normal, que haya sido originado por una condición de alarma. Este tipo de sentencias son tratadas de forma especial por los intérpretes del lenguaje, de forma que entre cada dos instrucciones ejecutadas se consulta un bit para comprobar si se ha producido alguna condición anómala. Cuando se produce alguna de estas condiciones se cambia el flujo normal de instrucciones y se salta a una rutina que determina el origen de la interrupción. Al terminar el tratamiento de la interrupción se vuelve al programa normal.

En el lenguaje BASIC hay varias sentencias que permiten el cambio de secuencia de ejecución de instrucciones, al producirse algunas condiciones externas. Las más comunes en la mayoría de las implementaciones de los lenguajes son:

- STRIG: Se produce al activar alguno de los pulsadores externos conectados a la tarjeta de adaptación de juegos.
- KEY: Se activa al pulsar una determinada tecla de función y alguna otra de movimiento del cursor.
- TIMER: Al terminarse un determinado período se cede control al programa referenciado en la sentencia.

Este tipo de interrupciones podríamos denominarlo "macroscópico", pues el comportamiento aparente es como el de una interrupción, pero internamente hay una exploración real. Veremos más adelante cómo se emplean las interrupciones de forma real, con la intervención de los circuitos internos del ordenador personal, para alterar el flujo de la ejecución de instrucciones y atender alguna tarea prioritaria.

Si conectamos a una entrada correspondiente al pulsador de la tarjeta de adaptación de juegos una entrada multiplexada de las señales de los sensores podremos recibir control al producirse una condición de alarma.

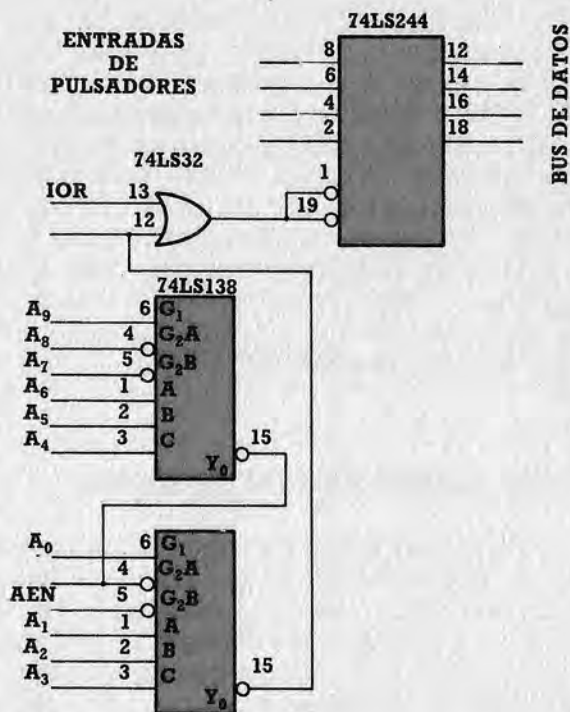


Fig. 4. Conexión a través del adaptador de juegos del IBM PC.

## Programa de control de alarmas

Los fenómenos que se producen en el sistema o recinto que queremos controlar aparecen de forma asíncrona con el programa que realiza el control. La generación asíncrona significa que puede producirse en cualquier instante y de forma independiente a lo que esté realizando el programa de control. Por esta razón el mecanismo que permita la captación del suceso deberá ser capaz de mantener la condición de activación durante determinado tiempo o activarse en el instante en que el fenómeno se produzca.

Una forma sencilla de consultar el estado de cada sensor consiste en explorar periódicamente cada uno de ellos. El circuito para realizar la consulta es sencillamente un multiplexor que seleccione la entrada a consultar y permita la lectura de su estado. ¿Qué ocurre si el estado de uno de los sensores se activa de forma intermitente? El circuito multiplexor solamente transmite el estado del sensor en el momento de la consulta y puede ocurrir que el instante de la consulta no coincida con el de la activación. Podemos adoptar dos formas de resolver el problema:

— Almacenar el estado activado, de forma que la indicación de activación se man-

tenga aunque pasado un cierto tiempo desaparezca la condición.

— Generar una interrupción al sistema, para notificarle que algo ha cambiado en el estado de algún sensor.

Las dos formas tienen interés práctico en la realidad, pues existen diferentes situaciones que hacen recomendable cada solución.

Para medidas que por su propia naturaleza presentan un estado activado estable o que poseen una cierta inercia en su activación, el método de exploración es apropiado, siempre que el intervalo entre muestras sea suficientemente corto, para que no pueda producirse la desactivación real de la causa. El problema asociado al muestreo periódico es que el ordenador no puede estar haciendo otra cosa que contar el paso del tiempo y una vez llegado el momento de las lecturas realizar la exploración de las señales oportunas.

Para medidas que tienen máxima prioridad y cuyos efectos pueden resultar muy perjudiciales, convendrá utilizar la técnica de interrupción, para efectuar correcciones o comunicación de alarmas en el menor tiempo posible. Este método tiene además la ventaja de que el ordenador puede estar haciendo otras cosas y, sin embargo, permanecer vigilante ante cualquier suceso externo.

Una solución intermedia entre las dos expuestas consiste en utilizar un circuito temporizador para indicar al ordenador cuándo tiene que realizar las operaciones de medida. De esta forma puede emplearse el ordenador para hacer cualquier otro tipo de cálculo y llegado el momento de efectuar las medidas, detener el proceso en curso y lanzar la exploración.

```

10 REM PROGRAMA SENCILLO DE CONTROL DE
 ALARMAS
20 REM UTILIZA LECTURA EN BUCLE DEL PUER-
 TO DE ENTRADA
30 REM CON LAS ENTRADAS DE ALARMA MUL-
 TIPLEJADAS
40 REM SUPONE ALARMAS ACTIVAS EN 1
50 REM PUERTO DE ENTRADA SELR0
60 REM PUERTO DE SELECCION SELW0
70 SELR0=&H200: SELW0=&H200
75 MAXENTRADAS=8
80 CLS: KEY OFF
90 PRINT "PROGRAMA DE MUESTREO DE ALAR-
 MAS"
100 FOR J=1 TO 1000
110 FOR I=0 TO MAXENTRADAS
120 OUT SELW0,I
130 A=INP(SELR0)

```

## EL TALLER DEL HARDWARE

```
140 IF A<>0 THEN PRINT "ALARMA EN:";I;
 "CODIGO=";$HEX(A)
150 NEXT I
160 NEXT J
```

La forma más habitual de incluir en un programa las alarmas será mediante las sentencias de interrupción como STRIG, si se cablean las señales a las entradas utilizadas por la sentencia de los pulsadores de juegos.

```
10 REM PROGRAMA DE CONTROL DE ALARMAS
POR STRIG
20 REM UTILIZA LAS ENTRADAS DE LA TARJE-
TA DE CONTROL DE JUEGOS
50 REM PUERTO DE ENTRADA &H201
70 SELR0=&H201
80 CLS: KEY OFF
90 PRINT "PROGRAMA DE PRESENTACION DE
ALARMAS"
100 ON STRIG(0) GOSUB 1000: REM SIMULA BO-
TON A1 EN 0
110 STRIG(0) ON
120 FOR J=1 TO 1000: REM PROGRAMA EN BU-
CLE
130 NEXT J
200 END
1000 REM ENTRADA POR INTERRUPCION
1010 A=INP(SELR0)
1020 PRINT "ALARMA EN:";$HEX(A)
1030 RETURN
```

Un punto a tener en cuenta y que resulta de suma importancia es la dependencia del tiempo del concepto alarma. En una instalación cualquiera el estado de las puertas, ventanas, consumos, temperaturas, etc., depende de la utilización normal que del sistema se esté realizando. A cada hora corresponde un estado de "normalidad". Por tanto, las condiciones de detección de alarma dependen de la hora y, por ello, en un sistema controlado por ordenador deberá ser posible el cambio de las condiciones que identifican las alarmas. Incluso la fecha deberá ser tomada en cuenta para la sensibilidad del sistema a la producción de alarmas.

Para una correcta instalación de un sistema de alarmas controladas por un ordenador personal es conveniente la división en sectores que puedan ser controlados de forma homogénea. Cada zona se representará en la pantalla por los diferentes estados que pueda adoptar a lo largo de los diferentes períodos con objeto de que el operador conozca no sólo que el sistema está correctamente, sino también cuál es su margen de maniobra. Es más fácil seguir el funcionamiento de la instalación si se está familiarizado con lo que tiene que hacer en cada instante.

Un área muy importante en la que el control mediante ordenador puede ser la pieza clave es en el aprendizaje de la operativa del sistema. Mediante un programa de simulación apropiado pueden producirse situaciones anómalas de forma ficticia, demandando del operador la toma de acciones que una vez introducidas en la máquina mostrarán el efecto producido. El entrenamiento de quién deba controlar el sistema puede así realizarse en condiciones "casi reales", con la consiguiente mejora en los resultados.

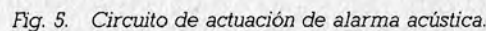
### ■ Actuadores en caso de alarma

Una vez que el sistema ha detectado una situación anómala es necesario que adopte las medidas correctivas adecuadas. La primera será guardar la información más completa posible de las circunstancias en las que se produce la situación de alarma, para poder determinar su origen. En segundo lugar deberá notificar al operador o a quien sea responsable, la producción de una condición que debe ser corregida. En tercer lugar deberán tenerse programadas las soluciones de corrección más apropiadas a cada caso y utilizando los medios al alcance de las posibilidades del sistema.

El almacenamiento de las condiciones en las que se produce la situación de alarma consistirá en guardar en algún medio permanente el conjunto de variables más significativas que definan el estado del sistema: situación de los conmutadores e interruptores, hora, dispositivos activos, etc. El modo más corriente será la producción de un listado con las anomalías detectadas, pero si se dispone de un medio de almacenamiento magnético podrán consolidarse los resultados de anomalías de diferentes días y tomar las medidas correctivas a largo plazo.

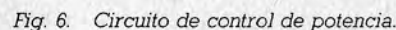
La notificación al operador o responsable del sistema se hará a través de un dispositivo acústico o bien mediante un mensaje codificado para su transmisión hasta la consola. Una alarma luminosa suele acompañar a la alarma acústica en la mayoría de las aplicaciones. En aplicaciones complejas de gran número de señales a controlar, será muy conveniente disponer de una consola con un diagrama sinóptico de los estados de los diferentes equipos bajo control. Si el sistema está centralizado en un ordenador, la representación en la

homogéneos permitirá el seguimiento de cada uno de ellos en una pantalla diseñada de forma adecuada para cada situación.



de añadirse otra etapa, hasta conseguir el nivel adecuado.

Las medidas correctivas son muy dependientes del sistema, siendo en general la activación de algún dispositivo de potencia para la actuación de un mecanismo de control y, por tanto, actuado por un circuito de control de potencia como el de la figura.



circuito que se coloca en paralelo con el Triac, formado por una resistencia y un condensador en serie, sirve para prevenir las subidas rápidas de tensión en las bornas de dicho Triac, que le podrían activar durante un ciclo de la tensión de red.

El equipo alimentado con este circuito se colocará en las bornas indicadas con CARGA. Este circuito puede servir para alimentar una bombilla o una resistencia de caldeo de hasta 300 vatios.

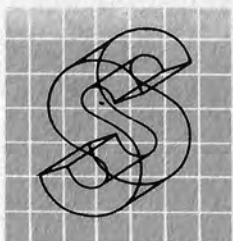


# APRENDER CON EL ORDENADOR

## NATURALEZA Y TECNOLOGIA

### FISICA

#### Movimiento Circular Uniforme



Se denomina así, al movimiento de un móvil que siguiendo una trayectoria circular recorre arcos de circunferencia iguales en tiempos iguales. Por esta circunstancia su velocidad es constante en módulo, pero

no en dirección ni sentido.

La posición  $P$  que ocupa el móvil en un instante dado viene determinada por el ángulo  $A$  (expresado en radianes) o por el arco  $S$  (expresado en metros). El sentido de giro positivo se considera el contrario a las agujas del

reloj, mientras que el sentido negativo es el de las agujas del reloj.

$$A=S/R$$

Donde  $R$  es el radio de la circunferencia descrita por el móvil en su movimiento.

En este movimiento se define la velocidad angular ( $W$ ) como el ángulo descrito por el móvil por unidad de tiempo (normalmente viene expresado en radianes/segundo).

Con el programa que a continuación proponemos, se podrán calcular velocidades circulares a partir de espacios recorridos y tiempos empleados, así como espacios o tiempos a partir de la introducción de otras variables. Los resultados se darán al final en varias unidades.

```
10 REM *****
20 REM ** MOVIMIENTO CIRCULAR **
30 REM *****
40 LET P=3.14159
50 CLS
60 PRINT "***** MOVIMIENTO CIRCULAR *****"
70 PRINT:PRINT:PRINT:PRINT
80 PRINT "A) CALCULO DE LA VELOCIDAD A PARTIR DEL ESPACIO RECORRIDO Y DEL TIEMPO"
90 PRINT:PRINT
100 PRINT "B) CALCULO DEL ESPACIO RECORRIDO A PARTIR DE LA VELOCIDAD Y DEL TIEMPO"
110 PRINT:PRINT
120 PRINT "C) CALCULO DEL TIEMPO A PARTIR DEL ESPACIO RECORRIDO Y DE LA VELOCIDAD"
130 PRINT:PRINT
140 PRINT "D) FIN DEL PROGRAMA"
145 PRINT:PRINT
150 PRINT "PULSE OPCION ELEGIDA"
160 LET O$=INKEY$:IF O$="" THEN GOTO 160
170 CLS
180 IF O$="A" THEN GOTO 400
190 IF O$="B" THEN GOTO 700
200 IF O$="C" THEN GOTO 1000
205 IF O$="D" THEN END
210 GOTO 50
370 REM *****
380 REM ** CALCULO DE LA VELOCIDAD **
390 REM *****
```

## APRENDER CON EL ORDENADOR

```

400 INPUT "ESPACIO RECORRIDO EN METROS=";E
410 IF E<=0 THEN GOTO 400
420 PRINT
430 INPUT "TIEMPO EMPLEADO EN SEGUNDOS=";T
440 IF T<=0 THEN GOTO 430
450 PRINT
460 INPUT "RADIO DE LA CIRCUNFERENCIA EN METROS=";R
470 IF R<=0 THEN GOTO 460
480 CLS
490 LET S=E/R:LET W=S/T
500 PRINT "EL ESPACIO RECORRIDO ES ";S;" RADIANTES = ";S/2/P;" VUELTAS"
510 PRINT "EL TIEMPO EMPLEADO ES ";T;" SEGUNDOS"
520 PRINT "EL RADIO DE LA CIRCUNFERENCIA ES ";R;" METROS"
530 PRINT
540 PRINT " LA VELOCIDAD ANGULAR ES:"
550 PRINT
560 PRINT " -(EN RADIANTES/SEGUNDO)=";W
570 PRINT
580 PRINT " -(EN REVOLUCIONES POR MINUTO)=";W*30/P
590 PRINT
600 PRINT " -(EN CICLOS/SEGUNDO)=";W*.5/P
610 PRINT
620 PRINT " LA VELOCIDAD LINEAL ES:"
630 PRINT
640 PRINT " -(EN METROS/SEGUNDO)=";W*R
650 GOTO 2000
670 REM *****
680 REM ** CALCULO DEL ESPACIO **
690 REM *****
700 INPUT "VELOCIDAD ANGULAR EN RADIANTES/SEGUNDO";W
710 IF W<=0 THEN GOTO 700
720 PRINT
730 INPUT "TIEMPO EMPLEADO EN SEGUNDOS=";T
740 IF T<=0 THEN GOTO 730
750 PRINT
760 INPUT "RADIO DE LA CIRCUNFERENCIA EN METROS=";R
770 IF R<=0 THEN GOTO 760
780 CLS
790 LET S=W*T:LET E=S*R
800 PRINT "LA VELOCIDAD ANGULAR ES ";W;" RADIANTES/SEGUNDO = ";W*30/P;" R.P.M"
810 PRINT "EL TIEMPO EMPLEADO ES ";T;" SEGUNDOS"
820 PRINT "EL RADIO DE LA CIRCUNFERENCIA ES ";R;" METROS"
830 PRINT
840 PRINT " EL ESPACIO RECORRIDO ES:"
850 PRINT
860 PRINT " -(EN RADIANTES)=";S
870 PRINT
880 PRINT " -(EN METROS)=";E
890 GOTO 2000
970 REM *****
980 REM ** CALCULO DEL TIEMPO **
990 REM *****
1000 INPUT "ESPACIO RECORRIDO EN METROS=";E
1010 IF E<=0 THEN GOTO 1000
1020 PRINT
1030 INPUT "VELOCIDAD ANGULAR EN RADIANTES/SEGUNDO";W
1040 IF W<=0 THEN GOTO 1030
1050 PRINT
1060 INPUT "RADIO DE LA CIRCUNFERENCIA EN METROS=";R
1070 IF R<=0 THEN GOTO 1060
1080 CLS
1090 LET S=E/R:LET T=S/W
1100 PRINT "EL ESPACIO RECORRIDO ES ";S;" RADIANTES = ";S/2/P;" VUELTAS"
1110 PRINT "LA VELOCIDAD ANGULAR ES ";W;" RADIANTES/SEGUNDO = ";W*30/P;" R.P.M"
1120 PRINT "EL RADIO DE LA CIRCUNFERENCIA ES ";R;" METROS"
1130 PRINT
1140 PRINT " EL TIEMPO EMPLEADO ES:"
1150 PRINT
1160 PRINT " -(EN SEGUNDOS)=";T
1170 PRINT
1180 PRINT " -(EN MINUTOS)=";T/60
2000 REM *****
2010 REM ** FINAL DEL PROGRAMA **
2020 REM *****
2025 PRINT
2030 PRINT " PULSE UNA TECLA PARA CONTINUAR"
2040 LET Z$=INKEY$:IF Z$="" THEN GOTO 2040
2050 GOTO 50

```

## Modificaciones para otros ordenadores

Este programa es válido para IBMPC, MSX.

### COMMODORE:

```
160 GET O$:IF O$="" THEN GOTO 160
2040 GET Z$:IF Z$="" THEN GOTO 2040
```

### SPECTRUM:

```
205 IF O$="D" THEN GOTO 9999
```

## SOCIEDAD

### Geografía: Cabos y golfos de Europa

Uno de los accidentes geográficos más interesantes que se dan en la geografía terrestre son los golfos, y consecuentemente los cabos.

Los golfos se pueden definir como grandes entrantes de mar en la tierra.

Los cabos por su parte se podrían definir como entrantes de la tierra en el mar.

Europa es un continente situado geográficamente de la siguiente manera: limita por el norte con el mar del Norte, al sur con el mar Mediterráneo, al este con Asia y al oeste con el océano Atlántico.

Como se puede observar, es un continente que está rodeado en sus tres cuartas partes por mar, y en consecuencia tiene un gran número de estos accidentes geográficos.

Otros accidentes geográficos de estos tipos podrían ser los fiordos y las rías, pero se diferencian de los golfos en que los dos primeros son de mucho menor tamaño; los fiordos más importantes los tenemos en Noruega, mientras que rías importantes las tenemos en Galicia (España).

A continuación vamos a presentar un programa que nos presente dos opciones, cada una de las cuales nos presentará respectivamente los golfos y los cabos más importantes del continente europeo, así como el nombre del país en el cual están situados.

```
10 REM *****
20 REM ** PROGRAMA DE CABOS Y GOLFOS **
30 REM ** DE EUROPA **
40 REM *****
50 CLS
60 DIM A$(13):DIM B$(13):DIM D$(10):DIM E$(10)
70 FOR I=1 TO 13
80 READ A$(I)
90 READ B$(I)
100 NEXT I
110 FOR I=1 TO 10
120 READ E$(I)
130 READ D$(I)
140 NEXT I
150 CLS
160 PRINT " **** CABOS Y GOLFOS DE EUROPA **** "
170 PRINT:PRINT
180 PRINT " A.- CABOS DE EUROPA "
190 PRINT " B.- GOLFOS DE EUROPA "
200 PRINT:PRINT:PRINT
210 INPUT "QUE OPCION DESEA";C$
220 CLS
230 IF C$<>"A" AND C$<>"B" THEN GOTO 160
240 IF C$="B" THEN GOTO 450
250 REM *****
260 REM ** LOS CABOS DE EUROPA MAS **
265 REM ** IMPORTANTES SON **
270 REM *****
280 FOR I=1 TO 13
290 PRINT A$(I);TAB(50);B$(I)
300 NEXT I
310 DATA "CABO DE ROCA", "PORTUGAL"
320 DATA "KANIN NOS", "RUSIA"
330 DATA "FONTUR", "ISLANDIA"
340 DATA "CABO DE INDENES", "NORUEGA"
350 DATA "CABO NORTE", "NORUEGA"
360 DATA "CABO LAND'S END", "INGLATERRA"
370 DATA "CABO FINISTERRE", "ESPAÑA"
380 DATA "CABO DE CREUS", "ESPAÑA"
390 DATA "CABO BLANCO", "TUNEZ"
```

```

400 DATA "CABO SPARTIVENTO", "ITALIA"
410 DATA "CABO DE SKAGEN", "DINAMARCA"
420 DATA "CABO ANDREAS", "CHIPRE"
430 DATA "CABO MATAPAN", "GRECIA"
440 GOTO 620
450 REM *****
460 REM *** LOS GOLFOS MAS IMPORTANTES ***
470 REM *** DE EUROPA SON ***
480 REM *****
490 FOR I=1 TO 10
500 PRINT E$(I); TAB(50); D$(I)
510 NEXT I
520 DATA "FINLANDIA", "FINLANDIA"
530 DATA "RIGA", "RUSIA"
540 DATA "DANZIG", "POLONIA"
550 DATA "VIZCAYA", "ESPAÑA"
560 DATA "LEON", "FRANCIA"
570 DATA "VENEZIA", "ITALIA"
580 DATA "HAMMAMET", "TUNEZ"
590 DATA "SALONICA", "GRECIA"
600 DATA "ANTALYA", "TURQUIA"
610 DATA "BOTNIA", "SUECIA"
620 INPUT "DESEA CONTINUAR (S/N)"; T$
630 IF T$ <> "S" AND T$ <> "N" THEN GOTO 620
640 IF T$ = "S" THEN GOTO 150
650 IF T$ = "N" THEN PRINT "FIN DEL PROGRAMA"
660 IF T$ <> "S" AND T$ <> "N" THEN GOTO 620

```

## Modificaciones para otros equipos

### COMMODORE:

En las líneas donde aparezca la sentencia CLS sustituir las por las sentencias:

NUMERO DE LINEA PRINT CHR\$(147)G

### SPECTRUM:

60 DIM A\$(13,15):DIM B\$(13,15):DIM D\$(10,15):DIM E\$(10,15)

## MATEMATICAS

### ■ Derivada e integral

Como de todos es sabido, dos de las principales funciones en matemáticas son la derivación y la integración.

La derivada de una función  $y=f(x)$  en un punto coincide con la tangente geométrica trazada a la curva de ecuación  $y=f(x)$  por dicho punto.

Por su parte la integral se puede decir que es la operación inversa de la derivada, pues se define como: aquella función que verifica que  $F'(x)=f(x)$ , y recibe el nombre de primitiva de  $f(x)$ .

La integración cumple una serie de propiedades siendo las más importantes.

1. La integral de una suma de funciones es igual a la suma de las integrales de esas funciones.
2. La integral del producto de una constante por una función es igual a la constante por la integral de la función.

En el programa que se expone a continuación se podrán obtener los valores de las derivadas e integrales según se quiera, puesto que al principio del mismo se dan a elegir dos opciones según lo que se pretenda obtener.

```

10 REM *****
20 REM *** CALCULO DE LA DERIVADA DE UNA FUNCION EN UN INTERVALO ***
30 REM *** INTEGRAL DE UNA FUNCION EN UN INTERVALO ***
40 REM *****
50 CLS
60 PRINT " DEFINICION DE LA FUNCION "
70 PRINT:PRINT:PRINT
80 PRINT " DEFINE LA FUNCION "
90 PRINT " HAZLO PRECEDIDA DE: "
100 PRINT " 120 DEF FN Y(X)=..."
110 PRINT "CUANDO LO HAYAS HECHO TECLSEA RUN 120"
120 STOP
130 CLS
140 PRINT "TECLSEA EL NUMERO QUE PRECEDE AL CALCULO QUE QUIERAS REALIZAR "

```

```

150 PRINT " 1.- DERIVADA DE LA FUNCION "
160 PRINT " 2.- INTEGRAL DE LA FUNCION "
170 INPUT Z
180 IF Z<>1 AND Z<>2 AND Z<>3 THEN GOTO 170
190 GOSUB 1000*Z
200 INPUT " QUIERES HACER OTRO CALCULO (S/N)";X$
210 IF X$="S" OR X$="s" THEN GOTO 130
220 INPUT "QUIERES DEFINIR OTRA FUNCION (S/N)";G$
230 IF G$="S" OR G$="s" THEN GOTO 60
240 END
1000 CLS
1010 PRINT " INTRODUCE LOS LIMITES DEL INTERVALO A,B"
1020 INPUT A,B
1040 INPUT "NUMERO DE PASOS";N
1050 DIM X(N):DIM V(N):DIM D(N)
1060 LET H=(B-A)/N
1070 LET X(0)=A
1080 FOR I=1 TO N
1090 LET X(I)=X(I-1)+H
1100 LET Y(I)=FN Y(X(I))
1110 LET D(I)=Y(I)
1120 NEXT I
1130 LET H=4
1140 FOR K=1 TO 4
1150 FOR I=0 TO N-K
1160 LET D(I)=D(I+1)-D(I)
1170 NEXT I:NEXT K
1180 PRINT "INTRODUCE EL VALOR DE X (DENTRO DEL INTERVALO) DEL CUAL QUIERAS HALLAR LA DERIVADA"
1190 INPUT X
1200 LET P=(X-X(0))/H
1210 INPUT "NUMERO DE ORDENADAS ";P
1220 LET E=0
1230 LET W=1/H
1240 PRINT " ORDEN : ", "VALOR DE Y:"
1250 FOR K=1 TO N-P
1260 LET E=E+W*D(P)/K
1270 PRINT K,E
1280 LET W=-W
1290 GOTO 420
1300 RETURN
2000 CLS
2010 PRINT "INTRODUCE LOS LIMITES DEL INTERVALO A,B "
2020 INPUT A,B
2030 PRINT " NUMERO DE PASOS "
2040 INPUT N
2050 IF N=0 THEN GOTO 710
2060 LET H=(B-A)/N
2070 LET I=.5*(FN Y(A)+FN Y(B))
2080 LET X=A
2090 FOR J=1 TO N-1
2100 LET X=X+H
2110 LET I=I+FN Y(X)
2120 NEXT J
2130 LET I=H*I
2140 PRINT "VALOR DE LA INTEGRAL ";I
2150 PRINT " FIN DEL CALCULO "
2160 RETURN

```

## Modificaciones para otros equipos

### COMMODORE:

Sustituir las sentencias donde aparezca  
CLS por las sentencias:

NUMERO DE LINEA PRINT CHR\$(147)

### SPECTRUM:

250 GOTO 9999

## PARA LOS MAS JOVENES

### Jugando a la ruleta

En esta sección vamos a aprender algunos conceptos básicos de probabilidades, para luego verlos aplicados mientras jugamos a la ruleta.

Una de las definiciones más básicas en el cálculo de probabilidades, es la de experimento aleatorio, que se define como aquel cu-

## APRENDER CON EL ORDENADOR

yos resultados no podemos predecir de antemano, aunque lo repitamos un número indefinido de veces. Si aplicamos esta idea al juego de la ruleta, el experimento aleatorio sería el acto de lanzar la bola para ver el número que sale; es un experimento cuyo resultado no podremos predecir nunca, por mucho que repitamos el acto.

Imaginémonos ahora que hemos estado jugando a la ruleta durante un tiempo considerable, durante este tiempo habremos jugado un número definido de partidas en cada una de las cuales nos habrá salido un resultado (un número). Al conjunto de esos resultados es a lo que se llama en estadística variable estadística.

Para hallar la probabilidad de acierto en un experimento aleatorio, se aplica la siguiente fórmula:

$$P(A) = \frac{\text{CASOS FAVORABLES}}{\text{CASOS POSIBLES}}$$

Donde P(A) significa la probabilidad de que ocurra el suceso A.

A continuación se expone un programa en donde usted puede practicar estos conceptos estadísticos mientras disfruta jugando a la ruleta.

```

5 REM *****
7 REM *** JUGANDO A LA RULETA ***
10 REM*****
15 CLS
20 INPUT "CON CUANTO DINERO EMPIEZAS";P
25 CLS
30 INPUT "DESEAS JUGAR A UN NUMERO (S/N)";A$
35 CLS
40 IF A$="" THEN GOTO 40
50 IF A$="N" THEN GOTO 110
60 INPUT "A QUE NUMERO JUEGAS";N
65 CLS
70 INPUT "CUANTO DINERO JUEGAS";X
75 CLS
80 IF X<P THEN GOTO 120
90 PRINT "NO TIENES TANTO DINERO"
100 GOTO 70
110 LET X=X-P
120 INPUT "JUEGAS A PAR/IMPAR (S/N)";B$
125 CLS
130 IF B$="" THEN GOTO 130

```

```

140 IF B$="N" THEN GOTO 210
150 INPUT "PAR O IMPAR (1/2)";S
155 CLS
160 INPUT "CUANTO DINERO";Y
165 CLS
170 IF Y<P-X THEN GOTO 210
180 PRINT "NO TIENES TANTO DINERO"
190 GOTO 160
200 LET S=0:LET Y=0
210 INPUT "JUEGAS A PASA/FALTA (S/N)";C$
215 CLS
220 IF C$="" THEN GOTO 220
230 IF C$="N" THEN GOTO 260
240 INPUT "PASA O FALTA (1/2)";T
241 IF C$="N" THEN GOTO 270
245 CLS
250 INPUT "CUANTO DINERO";Z
260 LET T=0
265 IF A$="N" AND B$="N" AND C$="N" THEN GOTO 490
266 RANDOMIZE TIMER
270 LET A=INT(37*RND)
280 IF A=N AND A<>0 THEN GOTO 310
290 LET P=P-X
300 GOTO 330
310 LET P=P+35*X
320 LET B=0
330 IF A=INT(A/2) AND A<>0 THEN LET B=1
340 IF A/2<>INT(A/2) AND A<>0 THEN LET B=2
350 IF B=S THEN GOTO 380
360 LET P=P-Y
370 GOTO 400
380 LET P=P+Y
390 LET C=0
400 IF A<=18 AND A<>0 THEN LET C=2
410 IF A>18 AND A<>0 THEN C=1
420 IF C=T THEN GOTO 450
430 LET P=P-Z
440 GOTO 460
450 LET P=P+Z
460 PRINT "HA SALIDO EL NUMERO";A
470 PRINT "TIENES ";P;"PESETAS PARA APOSTAR"
480 IF P<0 THEN GOTO END
490 INPUT "DESEAS SEGUIR JUGANDO (S/N)";Q$
500 IF Q$<>"S" AND Q$<>"N" THEN GOTO 490
510 IF Q$="S" THEN GOTO 30
520 PRINT "EL JUEGO HA TERMINADO"

```

### Modificaciones para otros equipos

#### COMMODORE:

En las líneas 15, 25, 65, 75, 125, 215, 245, sustituir CLS por PRINT CHR\$(147).

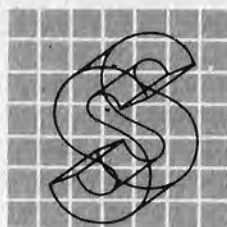
La línea 266 deberá suprimirse.

#### SPECTRUM:

480 IF P<0 THEN GOTO 9999

# PEQUEÑA HISTORIA DE LA INFORMATICA

## ■ El nacimiento del Univac, primer ordenador comercializado



SIEMPRE se ha creído que los norteamericanos fueron los primeros que comercializaron un ordenador. Sin embargo (aunque puede haber opiniones, ya que existen organismos que pueden considerarse en ciertos aspectos como privados), muchos consideran al Ferranti Mark I como el primer ordenador comercializado, y no precisamente en América, sino en Europa.

El Ferranti Mark I fue diseñado por un grupo de investigadores que trabajaban en la Universidad de Manchester. El primer modelo fue enviado al Royal Computing Machine Laboratory, en esa misma Universidad. Turing, ya por entonces muy conocido y considerado, escribió un programa para jugar al ajedrez contra la máquina. (Recordemos que mucho antes Leonardo Torres Quevedo había creado una máquina que jugaba al ajedrez, aunque realmente entre una y otra había diferencias enormes, siendo la de Torres Quevedo el embrión de lo que sería el Ferranti Mark I jugando al ajedrez.)

Otros ordenadores iban apareciendo poco a poco en los países europeos más industrializados. Sin embargo, los fines comerciales de muchas de estas máquinas no están muy claros, siendo utilizados muchos de ellos con fines científicos, en organismos estatales o paraestatales y otros muchos adquiridos por el Ejército de la nación donde se fabricaban. En Francia, por ejemplo, se fabricaron en los años

cincuenta la gama de los CUBA (Calculateur Universel Binaire) para el Ejército, y los GAMMA de Bull. Algo más tarde, aparecieron los CAB (Calculatrices Arithmétiques Binaires), que eran muy parecidos a los prototipos del CUBA. Los últimos modelos de CAB disponían de memorias de ferrita y eran en general máquinas muy completas, utilizadas casi siempre con fines científicos. En esta misma época, en Suecia (que tradicionalmente fabricaba registradoras de gran calidad) aparecieron otros modelos de calculadores, algunos de relés y otros de diferentes tipos, pero entre ellos el BARK (Binar Automatisk Relakalkyllator) fue el más difundido.

Volviendo al aspecto comercial de los ordenadores, el Ferranti no marcó un hito en la historia de la informática. Su repercusión no fue ni mucho menos tan importante como la del Univac I. Este ordenador descendía directamente del ENIAC (a pesar de sus enormes diferencias).

Eckert y Mauchly, siempre juntos, habían reorganizado su compañía, la Eckert-Mauchly Computer Corporation. La verdad es que eran fuertes de espíritu, porque consiguieron soportar todo tipo de dificultades. Habían tenido querellas y problemas con los principales constructores de ordenadores, habían soportado bancarrotas, habían recomprado su compañía e incluso Mauchly había tenido problemas con una Comisión a nivel estatal que se ocupaba de evitar atentados contra la seguridad del Estado. Pero eso no fue todo. A nivel técnico, los dos amigos-compañeros habían diseñado modelos de ordenador intermedios (de los que hoy no se habla apenas), cuyo funcionamiento era muy deficiente, perdiendo mucho tiempo, ilusiones, dinero... La "epopeya" había comenzado, como muchos años an-

## PEQUEÑA HISTORIA DE LA INFORMATICA

tes, con una enorme tarea a realizar: la confección de un censo, en este caso, el censo de los Estados Unidos de América de 1950. Y la pareja de científicos obtuvo un contrato del National Bureau of Standards (NBS) para desarrollar una máquina que fuera capaz de asumir esta tarea. Hasta el momento, todo parecía magnífico. Ellos eran conocidos, tenían prestigio, experiencia y un contrato. ¿Qué más podían pedir? Las circunstancias se encargarían de ir poniendo vallas en la trayectoria de estos dos investigadores.

Pero Eckert y Mauchly eran efectivamente fuertes, y estaban convencidos de que tarde o temprano acabarían sus problemas. En 1951 apareció el primer ordenador destinado a ser comercializado: el UNIVAC I. El Univac I era un sistema informático completo, no se trataba simplemente de un calculador universal. Además del procesador central, disponía de impresoras que funcionaban a velocidades increíbles para su época, lectoras de cintas magnéticas y una memoria externa. Su memoria era lo más conseguido dentro del conjunto. Era de líneas de retardo, pero también disponía de un sistema de almacenamiento a gran escala de banda magnética. El sistema permitía activar hasta diez bandas a la vez, lo que resultaba utilísimo, entre otras cosas, para labores de gestión, (No hemos mencionado que el sistema numérico que usaba el Univac era el decimal, y de ello hablaremos unas líneas más abajo. Sin embargo, es importante especificarlo, ya que la memoria estaba configurada teniendo en cuenta este dato.)

Las cien líneas de retardo de mercurio de la memoria podían almacenar 10 números, cuya longitud podía alcanzar los 12 dígitos. También disponía de unos "registros" o memoria intermedia, con doce líneas de retardo, muy útil. Efectivamente, la memoria era enorme, muy práctica para muchas tareas, pero resultaba muy lenta. (El tiempo de acceso era de varios cientos de milisegundos, haciendo los tiempos de lectura larguísimos.)

Antes hemos mencionado que el equipo se había diseñado para trabajar con números en notación decimal. Por esta razón, era mucho más voluminoso que, por ejemplo, el IAS, que era binario. Concretamente, el procesador central era unas diez veces más voluminoso que los de los ordenadores de la época, que funcionaban con dígitos binarios. Sus 8.000 tubos de vacío del procesador central ocupaban armarios de unos dos metros de alto por cinco de ancho. Su peso era de unas cin-

### ¿Sabía usted que...?

En uno de los laboratorios de la Universidad de Manchester se desarrolló un tipo de memoria que se llamará «memoria de tubo de Williams», en atención al doctor Williams, que dirigía el equipo investigador. La nueva memoria era inestable, almacenándose los bits en pequeñas zonas del tubo, que era necesario cada cierto tiempo regenerar, ya que, en caso contrario, la información se iba perdiendo poco a poco. Estas memorias tenían muchos inconvenientes, pero disponían de una característica utilísima: el acceso aleatorio. Cuando se dice que una memoria es de «acceso aleatorio» se está indicando que el tiempo de acceso a la información almacenada en ella es independiente de la dirección seleccionada (no importa si se desea acceder a la última dirección almacenada, o a la que se almacenó mucho antes, el tiempo de acceso siempre es igual).

El haz de electrones del tubo de estas memorias servía para la lectura y para la escritura, y podía orientarse a cualquier punto del tubo (localizando en el mismo tiempo cualquier dirección).

Estas memorias tuvieron poca vigencia, siendo desplazadas rápidamente por las memorias de núcleos de ferrita.

co toneladas. Imaginemos el sistema de refrigeración necesario para que la máquina funcionara con fiabilidad: era realmente enorme. Entre otros datos técnicos del procesador central diremos que tenía un ritmo de pulsaciones de 2.250.000 estados por segundo, realizando sumas de doce cifras en la mitad de tiempo que el Eniac: unos 110 microsegundos. Sin embargo, a pesar de sus 5.000 tubos de vacío, y de su enorme complejidad, fue la máquina más fiable hasta la década de los sesenta.

Uno de los puntos negros de esta nueva máquina era que apenas si disponía de medios para recibir información desde el exterior. Es verdad que tenía un teclado muy cómodo para introducir información grabándola en cintas, pero es fácil comprender que este sistema no es bueno en absoluto cuando la información a grabar es voluminosa. Y una de las ventajas del Univac I era que podía manejar grandes volúmenes de datos. El sistema era muy lento y aunque suponía una gran innovación, para grabar un dato, había que localizar en la cinta el lugar adecuado, y luego grabarla. Para ganar tiempo, el equipo disponía

de una memoria intermedia de registros, con la que podían acelerarse muchísimo los procesos. Estas memorias intermedias se revelaron eficacísimas, y con el tiempo formaron parte de la mayoría de las memorias de los equipos de otras compañías. En cuanto a la grabación de los datos en cintas, también fue un éxito, y un sistema a copiar. Sin embargo, la introducción de los datos por el teclado no tuvo la utilidad que predecían los diseñadores. Tan incómoda resultaba la introducción de la información por este sistema directo que, de hecho, fue abandonado durante muchos años, y no volvió a utilizarse hasta que empezó a desarrollarse el sistema de trabajo con el ordenador a «tiempo compartido».

El Univac I disponía también de un dispositivo para grabar en las cintas la información que se encontraba codificada en las tarjetas perforadas. Pero, aunque nos pueda parecer incomprensible, el dispositivo diseñado por Eckert y Mauchly sólo aceptaba tarjetas del modelo de IBM (80 columnas) y no podía adaptarse a las de 90 columnas de Remington Rand. (La Remington llegó a comprar la compañía Eckert y Mauchly y terminó el proceso de fabricación del Univac I.)

En los primeros modelos, los resultados se presentaban en unas «impresoras-máquinas de escribir», muy lentas, y poco prácticas, por lo que Eckert y Mauchly pusieron el máximo interés en desarrollar una buena impresora. Lo consiguieron, efectivamente, logrando para el Univac una impresora absolutamente revolucionaria sólo dos años más tarde (1953). Esta impresora fue la primera de una serie de alta velocidad que han ido desarrollándose desde entonces. Imprimía 600 caracteres por minuto, y podía escribir hasta 120 caracteres por línea.

Cuando Eckert y Mauchly comenzaron a vender sus ordenadores probablemente sus problemas sólo fueran ya un recuerdo lejano. Habían intentado hacer las cosas lo mejor posible, desde el principio, y durante años no habían tenido suerte (o al menos habían tenido que batallar contra mil pequeños detalles para prepararse a recibir el premio de la suerte).

En los comienzos del proyecto, el grupo Eckert-Mauchly pidió asesoramiento a los mejores matemáticos, físicos e ingenieros relacionados con el diseño y fabricación de calculadores universales. Entre la larga lista figuran Howard Aiken, Stibitz, Von Neumann, etc. A ninguno pareció interesarle demasiado la idea. Aiken pensaba que no era posible rentabilizar el proyecto, y que este tipo de má-

quinas eran magníficas para grandes organismos, pero nunca para empresas. Estas no dispondrían de medios para encargarlas. Además, la idea de almacenar programas tampoco le parecía tan importante. Von Neumann, por su parte, no tenía especial simpatía por sus patrocinadores, y consideraba que era tirar el dinero el diseñar una máquina que era prácticamente idéntica al EDVAC. ¿Qué es lo que aportaba la nueva máquina sobre el EDVAC? Prácticamente nada.

Pero Eckert y Mauchly eran de acero. Y estaban empeñados en continuar. Se buscaron un financiero, y siguieron con el proyecto. Sin embargo, en este caso no pudieron cumplir los tiempos previstos para que su prototipo funcionara de forma mínimamente fiable y comprobada. Y los costes seguían subiendo de forma exponencial. Estaban casi arruinados y necesitaban más dinero. El banco que les financiaba hasta el momento había sido comprensivo, pero ¡el banquero que les apoyaba murió en un accidente de aviación!

(Recuerdo a los lectores que esto no es un folletín como los que emite nuestra televisión estatal con frecuencia: son hechos reales, aunque explicados en pocas líneas puedan parecer hasta ridículos.)

Después del abandono de la familia del financiero, Eckert y Mauchly se vieron en una situación apuradísima. Habían invertido mucho dinero y esfuerzo y era vital continuar. En 1950 su compañía pasó a formar parte de la Remington Rand, que creó una división especial, la Univac Division, para continuar con el desarrollo del proyecto y posteriormente vender los ordenadores tanto a organismos estatales como privados. Más tarde esta compañía se fusionó con Sperry para formar una división aparte. Tras posteriores uniones, la heredera de Univac actualmente se llama Unisys.

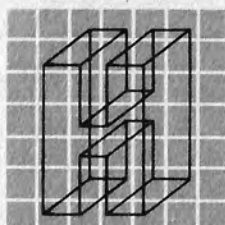
El Univac I se hizo muy famoso en los Estados Unidos, ya que pronosticó la victoria de Dwight Eisenhower en las elecciones presidenciales de 1952. A partir de ese momento los ordenadores se hicieron populares, organizándose debates en la televisión sobre el futuro de estas enormes y carísimas máquinas. La verdad es que el futuro ha sido muy diferente de lo previsto. Las enormes y carísimas máquinas que iban a dejar sin empleo a multitud de personas, han quedado obsoletas en pocos años. Los precios han bajado hasta extremos inimaginables. Los tamaños de las máquinas se han reducido espectacularmente, utilizándose actualmente microscopios para

## PEQUEÑA HISTORIA DE LA INFORMATICA

poder ver algunos de sus componentes. En lo que sí han acertado (sólo en cierto sentido) los pronósticos es que en el número de parados que iban a producir las nuevas máquinas. Es muy cierto que han creado mucho empleo, pero también han ayudado a subir la produc-

tividad a cotas tan altas que han provocado desajustes internos en la sociedad que tardarán años en equilibrarse. El futuro, sin embargo, es siempre el mismo: cambios, desequilibrios, ajustes, equilibrios. ¿Qué cambios revolucionarios surgirán en los ordenadores del futuro?

## Ordenadores vectoriales II



EMOS visto anteriormente la estructura de la multiplicación con coma flotante dentro de un ordenador vectorial. Recordemos cómo se rebajaban los tiempos de ejecución de las operaciones, aprovechando las estructuras pipeline y que el tiempo de arranque del pipeline no dependía en absoluto de la longitud de los vectores. Veamos ahora una nueva magnitud obtenida con dos valores: uno será el tiempo de ejecución utilizando un ordenador escalar (arquitectura de Von Neumann), y otro el de ejecución de la misma operación a través de un pipeline. Recordemos

que este tiempo (en ciclos) pasaba de  $4l$  a  $3 + 1$  (siendo  $l$  la longitud de los vectores o número de elementos del conjunto "vector"). La nueva magnitud que vamos a considerar es el "coeficiente de aceleración", que será la relación entre ambos, es decir,  $4l/3 + 1$ . En la figura que damos a continuación puede verse la curva correspondiente al coeficiente de aceleración, que indica bastante bien las características esenciales de los ordenadores vectoriales. En ella puede verse que en el infinito la curva alcanzaría el valor 4, es decir, en un ciclo de un ordenador vectorial se realiza una tarea que tardaría cuatro, en un ordenador escalar (naturalmente añadiendo la inicialización): asimismo, se ve que la mejora (aceleración producida al aumentar la longitud de los vectores va amortiguándose, a medi-

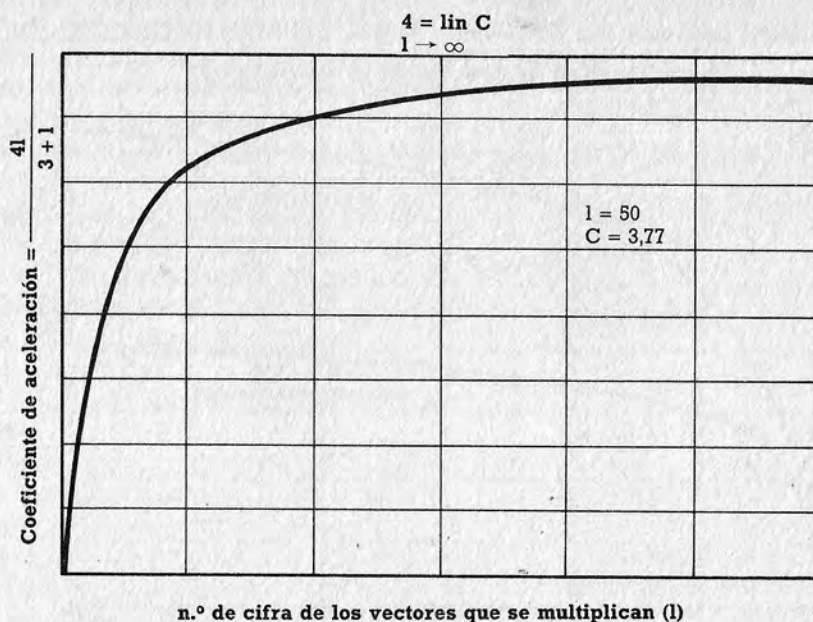


Fig. 1. Evolución de la eficacia del pipeline.

da que se va acercando la curva a la asíntota, es decir, a medida que va aumentando su longitud).

La curva se acerca a la asíntota bastante rápidamente. Para 50, ya está en el 95 por ciento del valor máximo. Queda, pues, claro que estas estructuras pipeline son muy útiles a partir de longitudes de vectores relativamente pequeñas. Sin embargo, como se ve, el arranque afecta mucho más en vectores de longitudes pequeñas, por lo que aquellos ordenadores que trabajen con vectores de pocos componentes deberían tener tiempos de arranque menores que aquellos cuyos trabajos afecten a vectores de longitudes grandes, donde este valor apenas si afectará la operación.

Pero no nos engañemos. A pesar de que estamos hablando de 2, 3, 4 ciclos, es importante indicar que ordenadores que disponen de tiempos de arranque cortos pueden tener de 25 a 30 ciclos improductivos y a partir de estas cifras se encontrarían todos los demás con valores superiores. Efectivamente, en la inicialización debe contarse además con las pérdidas de otros ciclos además de los debidos a los arranques de las estructuras pipeline.

Veamos ahora otras curvas que indican las características del ordenador: las correspondientes a dos estructuras pipeline sin encadenar, y a dos estructuras pipeline encadenadas. En ellas podemos apreciar cómo aumenta el coeficiente de aceleración para las estructuras encadenadas, aumentando las posibilidades casi al doble. Los valores son: 81

para la realización de las operaciones en un ordenador escalar,  $6 + 21$  (la misma curva anterior) P, para dos estructuras pipeline sin encadenar, y  $7 + 1$  para dos estructuras pipeline encadenadas. Los valores de los coeficientes serán, por tanto,  $81/6 + 21$  y  $81/7 + 1$ .

En el análisis del aumento de la eficacia de una matriz de pipelines se estudia habitualmente el encadenamiento de dos estructuras pipeline, multiplicación-suma, porque este tipo de operación es muy frecuente en los cálculos, y por ello tiene importancia en la aceleración de tareas con ordenadores vectoriales. Sin embargo, todas las operaciones no pueden encadenarse, por lo que el coeficiente de aceleración generalmente no coincide con los valores de la curva superior de la figura 2. La curva que indica el coeficiente real (no el teórico) se encontrará entre las dos mostradas, casi siempre más cerca de la inferior, dependiendo en cada tarea del número de operaciones que hayan podido encadenarse.

Recordemos la definición de "peak rate", como número de secuencias multiplicación-suma que cada ordenador es capaz de realizar en un segundo.

Este valor media en cierto modo las características de la unidad central, considerando que sólo se realizaban secuencias de operaciones encadenadas. Al calcularlo no se tenían en cuenta ni el tiempo de arranque ni la realidad de que muchas de las operaciones que se realizan no pueden encadenarse. Veamos, pues, otro concepto, el "megaflop resultante", que deriva del concepto de "peak rate",

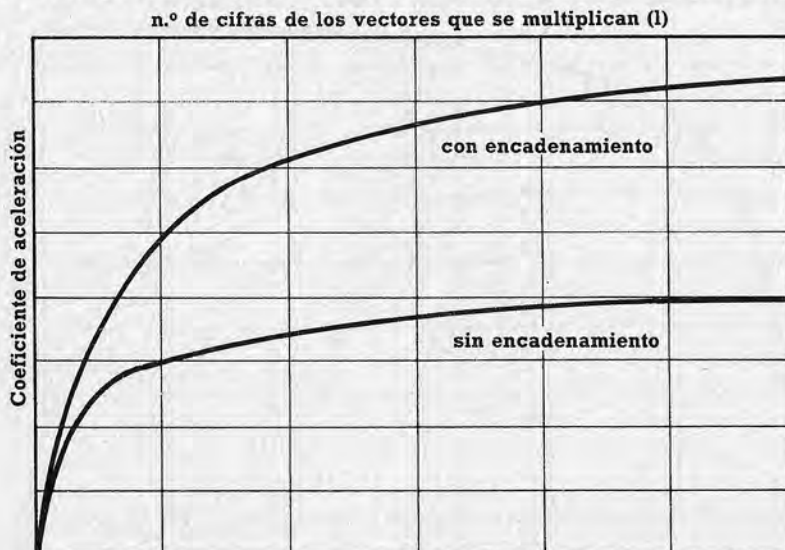


Fig. 2. Evolución de la eficacia con el encadenamiento de dos pipelines.

pero considerando todos los parámetros perturbadores que le afectan.

Duración del arranque del pipeline:

$$A \times C$$

Siendo A el arranque ("start np") y C el ciclo de base del ordenador.

Duración de las operaciones sin encadenar:

$$(0 - 0c) \times C \times 1$$

Siendo 0 el número total de operaciones a efectuar sobre un par de valores y 0c el número de operaciones encadenadas.

Duración de las operaciones encadenadas:

$$(0c \times C \times 1)/2$$

por tanto,  $t = A \times C + (0 - 0c/2) \times c \times 1$

El megaflop resultante será:

$MR = 0 \times 1 \times 10^3 / C (A + 1 (0 - 0C/2))$ . Atendiendo a la definición de "megaflop máximo" o "peak rate",  $M = 0/C$ , puede verse cómo el megaflop resultante se ve afectado.

Consideremos el paso siguiente. La unidad de proceso del ordenador necesita acceder a la memoria para tomar los datos, y más tarde transmitirlos también a ella. Pero recordemos que existía un momento intermedio en el que se almacenaban los valores de los pasos entre una y otra operación en una memoria especial de registros vectoriales. La capacidad de estos registros es limitada, pero deben almacenar cada uno un vector. Durante el curso de una operación se van tomando pares de valores de los dos vectores que se encuentran en el registro, explorándose los dos registros al mismo tiempo. El tamaño de los registros se mide por su "número de elementos". Cada "elemento" son ocho octetos (sesenta y cuatro bits). Si un registro tiene una longitud de "sesenta y cuatro", significa que en él se pueden almacenar sesenta y cuatro palabras, cada una de ellas de sesenta y cuatro bits.

Pero ¿qué ocurre si lo que debe almacenarse en alguno de los registros sobrepasa su capacidad?

En primer lugar, si los registros deben almacenar información que no llega a completar el registro, no existe problema alguno, y el proceso seguirá su marcha normal. Sin embargo, si la información a almacenar sobrepasa la capacidad del registro, el sistema seguido es llenar éste con toda la información posible (número de palabras total que acepte); realizar a

continuación las operaciones en secuencia (no debe olvidarse el tiempo de arranque de la estructura pipeline) y a partir de ese momento ir obteniendo un resultado por ciclo. Cuando se haya utilizado el registro completo se pasará a volver a llenarlo con el resto de la información, para continuar con la operación. En este punto hay que considerar dos cosas: el ordenador se encarga de tomar el número de componentes del vector equivalente a la capacidad de los registros (por tanto, no es necesario tomar decisión alguna a este respecto), y debemos considerar un nuevo tiempo de arranque con la segunda carga de componentes en los registros. Estas tareas de llenado de registros y "recorte de vectores" son propias del compilador.

Como hemos visto, habrá que tener en cuenta el tiempo de arranque de las estructuras pipeline cada vez que se vuelven a cargar los registros. Veamos, pues, cómo afecta el tamaño de los registros a la rapidez del ordenador vectorial:

Consideremos tres valores:

FT (A) primer entero superior o igual a A (función techo).

AR (Tiempo de arranque de la recarga de los registros).

AG (Tiempo de arranque (start-up) general).

El número de veces que se efectúe AR vendrá dado por  $FT(1/R)$ , y el tiempo de ejecución de las operaciones de que hemos hablado antes en ciclos de máquina será el siguiente:

$$t + AG + AR \times FT(1/R) + Ic \times n/2 + (0 - 0c) \times 1/2$$

siendo R el tamaño de los registros.

Los megaflops resultantes serán:

$$MR2 = 0 \times 1 \text{ y } t \times C$$

$$MR2 = M \times n \frac{2AG + 2AR \times FT(1/R) + (2 - 0c/0)1}{0 \quad 0}$$

Como puede verse en la figura, la longitud de los vectores va quitando importancia a las pequeñas pérdidas de tiempo achacables a la nueva carga de registros.

De todos los valores que vamos revisando podemos observar que los vectores largos son mucho más ventajosos para la estructura vectorial del ordenador que los vectores cortos. Veamos cuál es la óptima longitud de vec-

## TEMAS MONOGRAFICOS DE VANGUARDIA

tor para cada ordenador. Estudiemos qué es lo que se considera «un vector corto». Evidentemente, la longitud de un vector corto siempre será inferior al tamaño del registro. Este será corto si

$$(AC + Lv) \times Cv \leq l \times Lv \times Ce$$

$$Lv = \frac{AC \times CV}{l \times Ce - CV}$$

Siendo AC el arranque en ciclos, Lv la longitud del vector a partir del cual ya no es útil procesarlo como tal y Cv el ciclo de base de la unidad vectorial, Ce el ciclo de fase de la unidad escalar y Lv la longitud del vector.

Para estudiar el caso límite, CV + CE, AC + 24, n + 3, LV + 12.

Veamos ahora otro concepto nuevo. El ancho de banda. Hemos visto que en los ordenadores vectoriales dotados de estructuras pipeline es fundamental que éstas estén siempre adecuadamente alimentadas de datos para que no se pierdan registros, pero éstos se cargan desde la memoria central. El ancho de banda mide la velocidad de paso de información desde la memoria a la unidad central de proceso. Este concepto considera la transferencia total, es decir, el paso de la memoria a los registros, o el contrario, de los registros a la memoria. Normalmente el valor de que se trate se expresa en megapalabras por segundo o megaoctetos/segundo (una palabra equivale a ocho octetos).

Como en los ordenadores vectoriales la economía de tiempo es un objetivo fundamental a conseguir, la transferencia de información entre la memoria y los registros se lleva a cabo en paralelo con la ejecución de otras tareas: en concreto, con la realización de los cálculos. Es pues, evidente que para obtener unas buenas características la velocidad de transferencia de la memoria debe ser adecuada para que no se produzcan «esperas» en las estructuras de la unidad de cálculo. Sin embargo, no habría ningún inconveniente en que la memoria «esperara» la llegada de los datos. Vemos, pues, que el parámetro anterior, la anchura de banda, es fundamental en los ordenadores vectoriales. En cualquier caso, siempre se procura que los valores intermedios no pasen por la memoria central, y sean utilizados inmediatamente en otras estructuras, reduciendo los ciclos de máquina al mínimo.

Una operación cualquiera implica la transferencia de tres palabras o veinticuatro bits. Si se trata de n x L operaciones, habrá que

multiplicar este valor por 24. Si los vectores que estamos utilizando tienen una longitud l, y estamos realizando una secuencia de n operaciones, el tiempo en ciclos de máquina para las transferencias memoria-registros será el siguiente:

$$tmr = \frac{24 \times l \times n \times \text{porcentaje} \times 10^{-6}}{\text{número de puertas mem reg} \times \text{número de megaoctetos por puerta}}$$

A partir de este dato se puede obtener el «peak rate» resultante, y el tiempo de ejecución. (Ver en el glosario «Porcentaje de vectorización».)

Volvemos a recordar otra vez que para sacar partido de las características especiales de los ordenadores vectoriales es fundamental conocer su arquitectura. Uno de los puntos básicos será utilizar lo menos posible la memoria, y utilizar, por el contrario, en todo lo posible los registros, considerando, pues, su número, tamaño, etc. En la práctica, existen compiladores «vectorizadores» que facilitan enormemente la labor. También pueden adquirirse en el mercado bibliotecas de subprogramas, creadas por especialistas, que aprovechan al máximo las cualidades de las distintas máquinas disponibles. Finalmente, el programador puede estudiar cuidadosamente la estructura de la máquina, y concebir el programa ajustándose lo más posible a las normas prácticas que le dé el manual de su ordenador.

Hemos indicado que todos los ordenadores vectoriales disponen además de órganos escalares. No existen tareas enteramente vectoriales, y ciertas operaciones se retrasarían y complicarían muchísimo si se efectuaran en la unidad vectorial. Habrá que contar, pues, con un órgano escalar dentro del ordenador vectorial, y sus características influirán en los resultados y tiempos finales. Al usuario le interesará cómo afecta esta unidad en el cálculo de los tiempos de ejecución, en los megaflops resultantes. Los resultados, incluyendo unidad escalar, se expresarán no en megaflops resultantes, sino en un híbrido que considere también la parte escalar, es decir, un valor de megaflops compuestos, que, naturalmente, es inferior al de megaflops resultantes (recordemos que la configuración escalar es más lenta que la vectorial). Es importante hacer notar que la parte escalar tiene un peso importante en este valor, y que, por tanto, en los ordenadores vectoriales las características de su parte escalar son fundamentales. Sin

embargo, se ha probado que el coeficiente de aceleración tiende siempre a un valor máximo, que no depende fundamentalmente del cociente de las dos velocidades vectorial/escalar. Sin embargo, cuanto mayor sea la vectorización, más tarde se alcanzará este valor. Por tanto, el coeficiente de aceleración se verá afectado positivamente para porcentajes de vectorización elevados (para un 60 por ciento de vectorización, aunque la unidad vectorial sea veinticinco veces superior a la escalar, el coeficiente no se verá muy afectado, aunque si la vectorización alcanzara el 90 por ciento, el coeficiente será de más del doble).

Si el coeficiente velocidad vectorial/velocidad escalar es menor que un cierto valor, afectará también muy positivamente al coeficiente de aceleración.

Resumiendo:

**a)** La velocidad de cálculo de la parte escalar de un ordenador vectorial tiene una influencia fundamental en la velocidad de cálculo final.

**b)** La velocidad vectorial tiene una enorme influencia en la velocidad de cálculo final para cotas de vectorización muy altas (80 y 90 por ciento), que no se dan con frecuencia. Para obtener mejores velocidades finales, esta velocidad vectorial debe ser equilibrada con la velocidad escalar ( $V_v/V_e$  debe ser un número pequeño).

**c)** Debe existir también un equilibrio entre los elementos funcionales del ordenador, para que ninguno de sus órganos detenga o disminuya la velocidad de los restantes. Los valores de **b)** influyen considerablemente en este equilibrio sobre todo la relación de velocidades, dado que las cotas de vectorización tan altas que hemos mencionado, no son siempre alcanzables.

**d)** Un posible sistema para hacer que la velocidad escalar influya menos en la velocidad global consiste en efectuar en paralelo la parte escalar y la parte vectorial. Aunque este tipo de paralelismo sería muy deseable, es francamente difícil de obtener.



# TERMINOLOGIA

**Ancho de banda.** En las transferencias de información entre la memoria central y los registros, se denomina anchura de banda al número de megapalabras por segundo (o megaoctetes por segundo) que pasan de la memoria a los registros, o viceversa, es decir, este valor indica la capacidad de transferencia de información total.

**Aplicación vectorizable.** Aplicación cuya estructura puede adaptarse a ser ejecutada en un ordenador vectorial. La vectorización supone fundamentalmente dos tareas: formular leyes que gobiernan el fenómeno estudiado bajo un formulismo matemático que permita acceder a una programación en que las partes de «mayor intensidad de cálculo» se encuentren en el interior de bucles y, posteriormente, desarrollar estos bucles para obtener conjuntos de datos coherentes, que puedan ser manipulados globalmente en forma de vectores.

**Aplicación vectorizada.** Dícese cuando una determinada aplicación ha sufrido las adaptaciones adecuadas (técnicas de vectorización) para poder ser ejecutada en un ordenador vectorial. (Existe una gran distancia a recorrer entre la aplicación vectorizable y la vectorizada. En ocasiones, las complicaciones son tan grandes que no merece la pena la vectorización.)

**Coeficiente de aceleración.** Relación entre el número de ciclos necesarios para ejecutar una operación determinada con un operando en un ordenador escalar respecto del número de ciclos necesario para efectuar la misma operación en un ordenador vectorial.

**Compilador vectorizador.** Compilador que realiza la «vectorización» de una aplicación para su procesamiento en un ordenador de arquitectura vectorial. Normalmente los compiladores vectoriales se utilizan en conjunción con bibliotecas de subprogramas que ayudan a utilizar al máximo las cualidades del procesador vectorial donde van a ejecutarse los programas.

**Megaflops.** Unidad de medida que indica la potencia de proceso de un ordenador paralelo. Viene dado por el número de secuencias multiplicación-adición que pueden ser efectuadas en cada segundo en el ordenador.

**Megaflops compuestos.** Medida de la potencia de la máquina de megaflops, obtenida teniendo en cuenta la capacidad complementaria de proceso de la parte escalar del ordenador. Dado que el procesador escalar es más lento que el vectorial, el número total de megaflops obtenibles en una máquina híbrida, considerando ambas partes (escalar y vectorial de la máquina) es menor que el número máximo de megaflops («peak rate») característico de la parte vectorial.

**Megaflops resultantes.** Número de megaflops obtenido como potencia de la máquina, cuando se tienen en cuenta en el cómputo todos los parámetros perturbadores del equipo (tiempos de arranque del proceso en las estructuras pipeline, falta de encañamiento de algunas operaciones, etc.).

**Porcentaje de vectorización.** Este porcentaje se mide en tiempos, considerando el

## TERMINOLOGIA

tiempo de ejecución de la aplicación de forma escalar y el tiempo de ejecución de la aplicación, una vez vectorizada. Para obtener este porcentaje se mide el tiempo de la ejecución de la aplicación tratada escalarmente, y el tiempo de la parte escalar de la aplicación una vez vectorizada, calculándose el porcentaje escalar total de la aplicación. (Es decir, si la aplicación considerada de forma escalar dura 100, y la parte escalar de la aplicación una vez vectorizada dura 30, el porcentaje de vectorización será  $100-30=70\%$ , es decir, será del 70% de la aplicación total).

**Recorte de vectores.** Proceso por el que el ordenador selecciona automáticamente el número de bytes de un vector que puede guardar en sus registros internos (cuando dichos vectores son más largos que los registros de que dispone el ordenador). Normalmente, la tarea de recorte de vectores se realiza de un modo automático mediante rutinas incluidas en el programa objeto en tiempo de compilación (es decir, el programa-

mador no necesita tener en cuenta este proceso). El recorte de vectores (junto con el proceso de llenado de los registros) es una de las tareas que más lastran la capacidad de proceso de un ordenador vectorial.

**Vector corto.** Es aquel vector cuya longitud es tal que sería tratado mucho más eficazmente por un ordenador escalar (o por la parte escalar de un ordenador vectorial). Un valor muy general para la longitud de un vector corto podría ser 12. La fórmula para cada máquina, considerando sus características, es:

$$L_v = \frac{AC \times CV}{1 \times C_e - C_v}$$

**Velocidad escalar.** Capacidad de cálculo del procesador escalar incluido en un ordenador vectorial. Se mide en MIPS (megainstrucciones por segundo).

**Velocidad vectorial.** Capacidad de proceso de la parte vectorial del ordenador vectorial; se mide en megaflops.

# VOCABULARIO

**Operating system.** Software que controla la ejecución de los programas de ordenador. (Ver **Sistema operativo.**)

**OR.** Operador lógico cuya propiedad es que si P y Q son elementos simples, P OR Q será verdadero si al menos uno de los dos elementos antes citados es verdadero. Su tabla de verdad es la siguiente:

| P | Q | P OR Q |
|---|---|--------|
| 0 | 0 | 0      |
| 0 | 1 | 1      |
| 1 | 0 | 1      |
| 1 | 1 | 1      |

**OR exclusivo.** Operador lógico cuya propiedad es que si P y Q son elementos simples, P XOR Q es verdadero si uno cualquiera de los elementos es verdadero, y es falso si ambos elementos son verdaderos o si ninguno lo es. Su tabla de verdad es la siguiente:

| P | Q | P XOR Q |
|---|---|---------|
| 0 | 0 | 0       |
| 0 | 1 | 1       |
| 1 | 0 | 1       |
| 1 | 1 | 0       |

**OR, puerta lógica.** Puerta lógica que implementa el operador lógico OR.

**Ordenación.** Procedimiento para organizar los elementos de un conjunto según un determinado criterio. Criterios frecuentes son

el numérico (en el caso de listas numéricas, por orden creciente o decreciente) y alfabético.

También se puede establecer otra ordenación considerando los códigos que representan a los caracteres (letras, signos de puntuación, números, etc.), y de este modo, será anterior a 3, y a su vez £ A, a F.

**Ordenación de un fichero.** Como se sabe, es usual la distribución de un fichero en «registros», que a su vez se subdividen en «campos». Dichos registros pueden de este modo ordenarse según un determinado campo. Dicho campo se dice entonces que es la «clave» de esa ordenación. Por ejemplo, en una biblioteca ésta puede estar ordenada por el campo «autores» o por el campo «clase» (clase de libro, por ejemplo, novela, poesía, etc.).

**Ordenada.** Una de las coordenadas cartesianas representada generalmente por la letra Y.

**Ordenador.** Término que indica un sistema o dispositivo para el proceso de datos. Suele indicar la propia máquina digital para el cálculo numérico o para la gestión de listas de caracteres. Existen también ordenadores analógicos.

**Ordenador de uso general.** Ordenador adecuado para ejecutar todo tipo de programas y tareas. En contraposición a «ordenador dedicado», creado específicamente

## VOCABULARIO

para una tarea concreta, como, por ejemplo, el control de una máquina de corte.

**Ordenador especializado.** También puede utilizarse el término ordenador dedicado. Ha sido diseñado para desarrollar una tarea específica. Estos ordenadores pueden programarse para realizar algunas tareas concretas para las que han sido pensados, pero nunca para cualquier tarea general. Ejemplos de este tipo pueden ser los pequeños ordenadores de ciertos coches, que controlan algunas operaciones, los que controlan el rumbo de embarcaciones, etc.

**Ordenador híbrido.** Ordenador especializado para el control de ciertos procesos, que está constituido por dispositivos analógicos y digitales. No son ordenadores muy comunes.

**Ordenador personal.** Microordenador de dimensiones pequeñas, utilizado para pequeñas tareas, no profesionales, y en ocasiones para juegos.

**Ordenador principal.** En una organización compleja en la que se dispone de varios ordenadores, ordenador cuya actuación no se limita al proceso y almacenamiento de los datos, sino que actúa como gerente de las tareas de los restantes ordenadores de la red. Existe, pues, una organización jerárquica dentro de la red, designándose a los restantes ordenadores con el término de "satélites". El ordenador principal puede ser grande, o simplemente un microordenador.

**Ordenador satélite.** Ordenador que forma parte de una red y realiza determinadas tareas concretas y distribuidas desde el ordenador principal.

**Ordenar.** Clasificar. Organizar los elementos de un conjunto de acuerdo con un criterio establecido. Generalmente, el conjunto es un fichero.

**Ordinograma.** (Véase **Organigrama.**)

**Organigrama.** Representación gráfica elemental de la estructura de un programa, considerando fundamentalmente las acciones primarias que éste debe realizar frente a las estructuras de los datos utilizados por éste.

**OS.** Abreviatura de Operating System. (Véase **Sistema operativo.**)

**Output.** Término inglés que significa salida. Puede indicar transferencias de datos al exterior, o a los periféricos que reciben datos del ordenador principal.

**Overdubbing.** Término inglés que indica la operación de grabar datos digitales, sin haber borrado previamente los grabados con anterioridad. Dichos datos se almacenan junto con los anteriores, y pueden leerse posteriormente las dos series de datos de forma síncrona.

**Overflow.** Término inglés que significa desbordamiento. En informática indica cuando se han sobrepasado valores o condiciones permitidas. Puede producirse un "overflow" si se intenta superar la capacidad de la memoria, o el número mayor que un registro puede aceptar, etc.

**Overlap.** Término inglés que significa superposición, solapamiento. Técnica que implica un cierto paralelismo para incrementar el rendimiento de la máquina. Se puede producir un overlap, por ejemplo, cuando se realiza la búsqueda de la siguiente instrucción, mientras se está ejecutando la anterior, etc. Si el solapamiento se realiza en procesos de operaciones aritméticas, el proceso se llama *pipelining*.

**Overlay.** Técnica de aprovechamiento de la memoria central que permite cargar en dicha memoria central distintas secciones de código de un programa, "recubriendo" algunas de las que se encontraban anteriormente y que ya no van a ser ejecutadas de nuevo. Se utiliza cuando es necesario procesar programas muy grandes, y no se dispone de memoria suficiente.

**Overload.** Término inglés que significa sobrecarga.

**Overwrite.** Término inglés que significa sobrescribir. En informática indica grabar datos sobre los que se encontraban anteriormente en esas posiciones de la memoria, destruyendo, por tanto, todo lo que se encontraba anteriormente en esas direcciones.

**PACK.** Término inglés que significa comprimir (a veces se traduce por «empaquetar»). Se refiere a la comprensión de los datos para ser almacenados en la memoria. Los datos se codifican en binario, mediante cuatro bits. Con este sistema, un byte puede contener dos cifras decimales, almacenando

do el signo en otro byte situado a la derecha de las cifras.

**Padding.** Término inglés que en informática se refiere al relleno de un bloque con datos sin interés.

**Paddle.** Mando de control. Dispositivo de entrada típico de ordenadores personales. Va provisto de un mando mediante el cual se pueden introducir al ordenador valores continuos, no digitales. Los valores que se obtienen son proporcionales al giro del mando.

**Page.** (Véase *Página*.)

**Página.** Una de las divisiones de la memoria principal de un ordenador. Esta puede considerarse como dividida, en bloques o «páginas». En ordenadores de 8 bits, una página está constituida por dos a la ocho (256 posiciones), resultando, por tanto, que 1K = 4 páginas. A nivel de software se manejan en ocasiones páginas mayores de memoria (hasta 64 Kbytes).

**Paginación.** Técnica por la cual se divide la memoria central de un ordenador en páginas. (Véase *Memoria virtual*.)

**Paging.** (Véase *Paginación*.)

**Palabra.** Unidad elemental de memoria. Corresponde a una instrucción simple de

lenguaje máquina manejable por el ordenador en una única operación. Esta está constituida por un conjunto de bits en los que está codificada la instrucción. En tiempos anteriores, las celdas de la memoria tenían el mismo número de bits que las instrucciones de máquina, indicando de este modo la potencia del ordenador de manera inequívoca. En los ordenadores modernos, estos dos valores no tienen por qué coincidir, siendo generalmente la palabra de lenguaje máquina de más bytes que la celdilla de memoria.

**Palabra clave.** Palabra reservada que el ordenador reconoce e interpreta. Puede ser un código de instrucción (READ), por ejemplo.

**Palabra reservada.** Puede tratarse de una palabra clave (véase **Palabra clave**) o puede ser una etiqueta o un nombre de variable, etcétera, que tenga fines de identificación. Estas palabras deben tenerse en cuenta a la hora de asignar nombres, para evitar duplicidades.

**Panel.** Parte de la consola del ordenador que contiene los controles e interruptores básicos.

**Pantalla.** Zona del terminal o monitor que sirve para el intercambio de información entre el usuario y el ordenador. (En la pantalla pueden aparecer, entre otras cosas, los resultados.)





